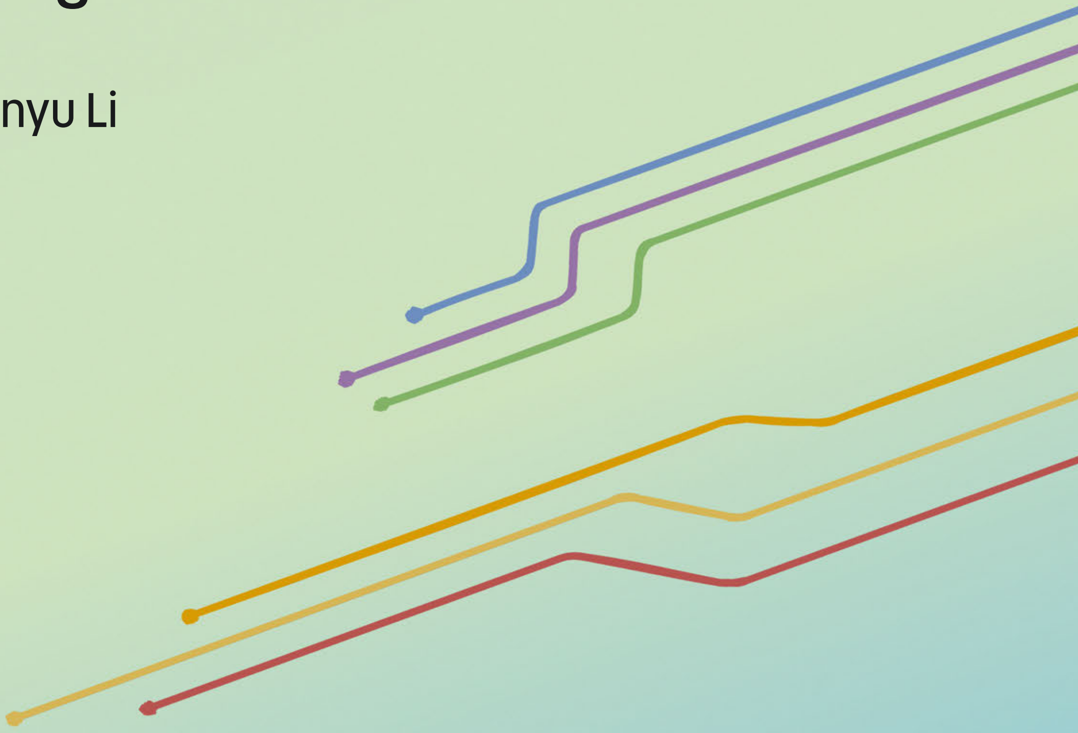
Linköping Studies in Science and Technology
Dissertation No. 2218

Ontology-Driven Data Access and Data Integration with an Application in the Materials Design Domain

Huanyu Li



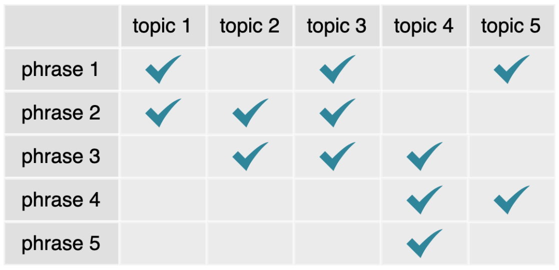
LINKÖPING
UNIVERSITY

Errata for "Ontology-Driven Data Access and Data Integration with an Application in the Materials Design Domain"

Huanyu Li
Linköping Studies in Science and Technology.
Dissertation No. 2218

Page 42. The second sentence of Section 4.1.2 should be updated as: $\mathcal{FL}_0$ allows atomic concepts, the universal concept, intersection and value restriction.

Page 89. Figure 6.4 (a) should be updated as follows (phrase 2 is also a representative phrase for topic 2).



	topic 1	topic 2	topic 3	topic 4	topic 5
phrase 1	✓		✓		✓
phrase 2	✓	✓	✓		
phrase 3		✓	✓	✓	
phrase 4				✓	✓
phrase 5				✓	

Page 134. The fourth sentence of the paragraph starting with "The **second** observation ..." should be updated as: UltraGraphQL and HyperGraphQL outperform other systems for some smaller datasets (e.g., UltraGraphQL's QETs of Q1 and Q2, HyperGraphQL's QETs for Q1 from 1K-1K to 4K-4K).

Page 135. morph-morphql should be updated as: morph-graphql.

Linköping Studies in Science and Technology

Dissertations, No. 2218

Ontology-Driven Data Access and Data Integration with an Application in the Materials Design Domain

Huanyu Li

Linköping University
Department of Computer and Information Science
Division of Database and Information Techniques
SE-581 83 Linköping, Sweden

Linköping 2022

Edition 1:1

© Huanyu Li, 2022
ISBN 978-91-7929-267-6 (print)
ISBN 978-91-7929-268-3 (PDF)
ISSN 0345-7524
DOI: https://doi.org/10.3384/9789179292683

Typeset using XeTeX

Printed by LiU-Tryck, Linköping 2022

Dedicated to all the teachers and supervisors I encountered through the journey of my studies!

书山有路，学海无涯，

积跬步以至千里，积小流以成江海！

得遇至亲、良师、益友，幸甚足矣！

ABSTRACT

The Semantic Web aims to make data on the web machine-readable by introducing semantics to the data. Ontologies are one of the critical technologies in the Semantic Web. Ontologies, which provide a formal definition of a domain of interest, can play an important role in enabling semantics-aware data access and data integration over heterogeneous data sources. Traditionally, ontology-based data access and integration methods focus on data that follows relational data models. However, in some domains, such as materials design, the models that data follows and the methods by which it is shared differ today. Data may be based on different data models (i.e., relational models and non-relational models) and may be shared in different ways (e.g., as tabular data via SQL queries or API (Application Programming Interface) requests, or as JSON-formatted data via API requests). To address these challenges, conventional ontology-based data access and integration approaches must be adapted. The recently developed GraphQL, a framework for building APIs, is an interesting candidate for providing such an approach, although the use of GraphQL for integration has not yet been studied.

In this thesis, we propose a GraphQL-based framework for data access and integration. As part of this framework, we propose and implement a novel approach that enables automatic generation of GraphQL servers based on ontologies rather than building them from scratch. The framework is evaluated via experiments based on a synthetic benchmark dataset. Further, we utilize the field of materials design as a target domain to evaluate the feasibility of our framework by showing the use of the framework for the Open Databases Integration for Materials Design (OPTIMADE), which is a community effort aiming to develop a specification for a common API to make materials databases interoperable. At the beginning of this work, no ontologies existed for the domain of computational materials databases. As our approach requires the use of an ontology, we developed one: the Materials Design Ontology (MDO). Furthermore, when new databases are added or new kinds of data are added to existing databases, the coverage of the ontology driving the GraphQL server generation may need to be enlarged. Therefore, we study how ontologies can be extended and propose an approach based on phrase-based topic modeling, formal topical concept analysis and domain expert validation. In addition to extending MDO, we also use this approach to extend two ontologies in the nanotechnology domain.

This work has been supported by the Swedish National Graduate School in Computer Science (CUGS), the Swedish e-Science Research Centre (SeRC), the Swedish Research Council (Vetenskapsrådet, dnr 2018-04147) and the EU project VALCRI (FP7-IP-608142).

POPULÄRVETENSKAPLIG SAMMANFATTNING

Vi lever i en värld som är full av data. Man kan säga att vårt dagliga liv styrs av data. År 2020 var volymen global data ca 64,2 zettabyte. Det förutspås att volymen global data skulle nå upp till 180 zettabyte år 2025. Data av detta slag består av information som utgörs av transaktioner från företag, forskning, sociala medier etc. Tillväxten är högre än tidigare eftersom vi befinner oss i covid-19-pandemin, och fler människor måste därför oftare arbeta och delta i aktiviteter online. På grund av utvecklingen av datorrelaterade teknologier kan vi producera data i ett stort antal olika sammanhang, analysera och lära av dessa data samt bygga upp datadrivna arbetsflöden. Till exempel kan man i materialdesigndomänen simulera extrema förhållanden för materialexperiment med datorprogram istället för att utföra experiment i ett riktigt labb. Även om data effektiviserar många aktiviteter i det dagliga livet och inom forskning, står vi inför utmaningen att data ibland inte är FAIR. FAIR-principerna anger att data ska vara sökbara (Findable), tillgängliga (Accessible), interoperabla (Interoperable) och återanvändningsbara (Reusable). Inom olika områden bedrivs forskning för att anpassa datahanteringen till dessa principer, inklusive inom materialvetenskap. Ontologier och ontologibaserade tekniker har erkänts möjliggöra dessa principer. Termen ontologi har sitt ursprung i filosofin, där det är namnet på läran om vad som är, om objektens typ och strukturer, egenskaper, förhållanden inom varje område av verkligheten. 1980 introducerades ontologier av Alexander et al. ur ett kunskapstekniskt perspektiv och har sedan dess spridit sig till många delfält inom datavetenskap. Ontologier kan intuitivt ses som en definition av de grundläggande termerna och relationerna för en intressedomän och reglerna för att kombinera dessa termer och relationer. Ontologier används för kommunikation mellan människor och organisationer genom att tillhandahålla en gemensam terminologi över en domän. Ontologier kan ge en delad standardiserad representation av kunskap om en domän. Genom att beskriva data med hjälp av ontologier blir data mer lätt att hitta (Findable). Genom att använda ontologier för att representera metadata kan tillgänglighetsnivån höjas (Accessible). Genom att använda samma terminologi som definieras av ontologier, möjliggörs interoperabilitet (Interoperable). Slutligen, eftersom ontologier delas och standardiseras, stöds återanvändbarhet (Reusable). För att göra data interoperabel och utbytbar behöver vi vanligtvis ett ramverk för att ge enhetlig och semantikmedveten tillgång till data över flera datakällor.

I denna avhandling presenterar vi ett GraphQL-baserat ramverk för dataåtkomst och integration med hjälp av en ontologi för att generera en GraphQL-server. GraphQL är ett nyutvecklat konceptuellt ramverk för att bygga API:er och kan stödja dataåtkomst och integration. GraphQL introducerar ett GraphQL-schema för att specificera vilken data som kan begäras; ett graffrågespråk som tillåter att skriva GraphQL-frågor; resolverfunktioner som får åtkomst till backend-datakällor och omstrukturerar data enligt schemat. Om GraphQL-schemat återspeglar semantiken för data från flera källor, och resolverfunktionerna kan hämta data från flera källor och strukturera data enligt GraphQL-schemat, kan heterogeniteten hos data från dessa källor behandlas. Vi föreslår och implementerar formella metoder för att automatiskt skapa GraphQL-servrar baserade på en ontologi. Vi tillämpar vidare detta ramverk inom materialdesignområdet. Inom ramen för detta ramverk fokuserar vi på att utveckla en ontologi för materialdesigndomänen och hur man utökar domänontologier. Vi utvecklar en ontologi för materialdesigndomänen (Materials Design Ontology, MDO), föreslår en metod för hur man utökar domänontologier och tar sedan fram kandidater som kan utöka MDO och två ontologier inom nanoteknikdomänen.

Acknowledgments

Back in 2015, when the LiU-HIT master exchange program brought me to Linköping, I could not have imagined having such an amazing journey in academical research. Thanks to the exchange program between the School of Software at Harbin Institute of Technology (HIT) and the Department of Computer and Information Science at Linköping University (LiU), I was given this unique opportunity to such a wonderful experience.

Without the support and help from the people around me throughout the writing of this thesis, I would not have been able to finish it. I am fortunate to have you all around me. I would like to start by expressing my deep gratitude to my principal supervisor, Professor *Patrick Lambrix*, for introducing me to the world of research. In spite of many challenges, your guidance, encouragement and words of support along the way give me the help and confidence to thrive forward. Your insightful feedback and advice pushed me to sharpen my thinking and brought my work to a higher level. Thank you so much, Patrick!

To my wonderful supervision team! I would like to thank Professor *Nahid Shahmehri*, for your great support, guidance and care. I am also grateful to Associate Professor *Rickard Armiento* and Associate Professor *Olaf Hartig.* Without your energy and time spent supervising and supporting me, I could not move this far. It is lucky among the lucky that I have all of you as my co-supervisors.

My sincere thanks go to the discussants in my 60% seminar for PhD, Associate Professor *Eva Blomqvist*, and Professor *Igor Abrikosov.* Thank you for the questions and discussions.

My special thanks go out to Patrick for nominating me for the Lawson Stipendium and to the IDA board for selecting me. It is indeed a great honor and encouragement to me. Without the international collaboration, I would not have been able to qualify for this prize. Therefore, I am grateful to *Catia Pesquita*, *Ernesto Jiménez-Ruiz*, *Daniel Faria*, *Valentina Ivanova*, *Zlatan Dragisic*, *Mina Abd Nikooie Pour*, *Ying Li* and many other co-authors from the ontology matching community.

I thank the colleagues at IDA's administrative department for the kind and helpful assistance you have provided me with. I would like to express my gratitude to *Anne* for your help. My thanks go to *Brittany* for your many constructive comments on this thesis. I also want to thank *Sijin*, *Kai* and *Peige* for proofreading some parts of this thesis.

I thank the colleagues at ADIT for making ADIT a pleasant place to work. Especially, I would like to thank Associate Professor *Niklas Carlsson*. Your questions during many presentations have been an inspiration for me.

I thank the Swedish National Graduate School in Computer Science (CUGS), the Swedish e-Science Research Centre (SeRC), the Swedish Research Council (Vetenskapsrådet, dnr 2018-04147) and the EU project VALCRI (FP7-IP-608142) for financial supports.

There has been a lot more to life in the past five years than just writing, coding, and studying. I thank Professor *Zebo Peng* and Professor *Nahid Shahmehri* for organizing and leading the badminton group. I have enjoyed the time with the group very much and have improved my badminton skills quite a lot. In addition, I want to take this opportunity to thank my friends, *Peige*, *Kai* and many others. My life is made more enjoyable by your presence. As the travels, games, and talks are precious and unforgettable, the memories will be as well.

Last but not least, I would like to thank my family, especially my parents, for your understanding and support of every decision that I have made. Thank you so much for encouraging me and for standing by me! 感谢一直以来支持我的父母、家人，你们是我前进路上最坚实的后盾！

This journey is about to come to a conclusion. Fun and challenges are always best friends along the path of going forward. As I am writing this, the past five years flashed over in my mind like a movie. With all the previous memories well kept in mind, it is also time to look forward to the new adventure!

*Huanyu Li/*李环宇

Linköping, 2022

Contents

List of Figures

List of Tables

List of Listings

List of Acronyms

AST	Abstract Syntax Tree
API	Application Programming Interface
CSV	Comma Separated Values
DNF	Disjunctive Normal Form
FAIR	Findable, Accessible, Interoperable, and Reusable
FTCA	Formal Topical Concept Analysis
GCI	General Concept Inclusion
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notation
JSON-LD	JSON for Linked Data
LinGBM	Linköping GraphQL Benchmark
LOV	Linked Open Vocabularies
MDO	Materials Design Ontology
OBDA	Ontology-Based Data Access
OBDI	Ontology-Based Data Integration
OBG-gen	Ontology-Based GraphQL Server Generation
OPTIMADE	Open Databases Integration for Materials Design
OWL	Web Ontology Language
REST	Representational State Transfer
RDF	Resource Description Framework
RML	RDF Mapping Language
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language

SQL	Structured Query Language
ToPMine	Topical Phrase Mining
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition

External Publications

Work included in the thesis

- **Paper I:** Huanyu Li, Olaf Hartig, Rickard Armiento, Patrick Lambrix, Ontology-Based GraphQL Server Generation for Data Access and Integration, 2022. (Submitted)

  The paper presents a GraphQL-based framework for data access and integration in which an ontology drives the generation of GraphQL servers. The need for such a system arises from discussions with OPTIMADE experts and researchers in the Swedish e-Science Research Centre. The author outlines the initial idea and the framework in the paper. The idea and the framework are further developed through discussions among the authors. The author is responsible for the prototype implementation and the evaluation. The paper is drafted by the author and is written together with the co-authors.

- **Paper II:** Patrick Lambrix, Rickard Armiento, Anna Delin, and Huanyu Li. "Big Semantic Data Processing in the Materials Design Domain." In: *Encyclopedia of Big Data Technologies*. Springer, 2019. DOI: `10.1007/978-3-319-63962-8_293-1`

- **Paper III (an extended version of Paper II):** Patrick Lambrix, Rickard Armiento, Anna Delin, and Huanyu Li. "FAIR Big Data in the

Materials Design Domain." In: *Encyclopedia of Big Data Technologies.* accepted. Springer, 2022

These two papers investigate the topic of big semantic data processing in the materials design domain with respect to the state of the art of databases, ontologies in the domain. The author is responsible for studying and analyzing existing ontologies in the materials design domain and for writing the ontologies relevant parts of the two papers.

- **Paper IV:** Huanyu Li, Rickard Armiento, and Patrick Lambrix. "An Ontology for the Materials Design Domain." In: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020.* Vol. 12507. Lecture Notes in Computer Science. Springer, Cham, 2020, pp. 212–227. DOI: `10.1007/978-3-030-62466-8_14`

  The paper presents the Materials Design Ontology. The need for such an ontology arises from discussions with OPTIMADE experts and researchers in the Swedish e-Science Research Centre. The author outlines the initial idea of the ontology. The scope and the requirements analysis of the ontology are defined through discussions among authors (the first co-author is a domain expert; the other co-author is an ontology engineering expert). The author is the main developer of the ontology with relevant deliverables supported through discussions with the co-authors. The domain expert reviews the content of the ontology while the ontology engineering expert reviews the logical basis of the ontology. The paper is drafted by the author and is written together with the co-authors.

- **Paper V:** Huanyu Li, Rickard Armiento, and Patrick Lambrix. "A Method for Extending Ontologies with Application to the Materials Science Domain." In: *Data Science Journal* 18.1 (2019). DOI: `10.5334/dsj-2019-050`

  The paper presents an approach for extending domain ontologies with applications in the nanotechnology domain. The author proposes the initial idea behind the paper. The idea and the framework are polished after discussions among authors. The author is responsible for the prototype implementation and the experiments. The paper is drafted by the author and is written together with the co-authors.

- **Paper VI:** Mina Abd Nikooie Pour, Huanyu Li, Rickard Armiento, and Patrick Lambrix. "A First Step towards Extending the Materials Design Ontology." In: *Proceedings of the Workshop on Domain Ontologies for Research Data Management in Industry Commons of Materials and Manufacturing (DORIC-MM 2021) co-located with the 18th European Semantic Web Conference (ESWC 2021).* 2021, pp. 1–11. URL: `http://purl.org/net/epubs/work/50300311`

  The paper presents the application of the approach presented in **Paper V** in the materials design domain. The author selects the relevant corpus and contributes to the evaluation and the writing of the paper.

- **Paper VII:** Huanyu Li, Rickard Armiento, and Patrick Lambrix. "Extending Ontologies in the Nanotechnology Domain using Topic Models and Formal Topical Concept Analysis on Unstructured Text." In: *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019).* Vol. 2456. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 5–8. URL: `http://ceur-ws.org/Vol-2456/paper2.pdf`

  The paper presents a poster of **Paper V**.

Other related publications

- Huanyu Li, Zlatan Dragisic, Daniel Faria, Valentina Ivanova, Ernesto Jiménez-Ruiz, Patrick Lambrix, and Catia Pesquita. "User validation in ontology alignment: functional assessment and impact." In: *The Knowledge Engineering Review* 34 (2019), e15. DOI: 10.1017/S0269888919000080

- Mina Abd Nikooie Pour, Huanyu Li, Rickard Armiento, and Patrick Lambrix. "A First Step towards a Tool for Extending Ontologies." In: *Proceedings of the Sixth International Workshop on the Visualization and Interaction for Ontologies and Linked Data co-located with the 20th International Semantic Web Conference (ISWC 2021).* Vol. 3023. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 1–12. URL: `http://ceur-ws.org/Vol-3023/paper2.pdf`

- Robin Keskisärkkä, Huanyu Li, Sijin Cheng, Niklas Carlsson, and Patrick Lambrix. "An Ontology for Ice Hockey." In: *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019).* Vol. 2456. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 13–16. URL: `http://ceur-ws.org/Vol-2456/paper4.pdf`

- Zlatan Dragisic, Valentina Ivanova, Huanyu Li, and Patrick Lambrix. "Experiences from the Anatomy track in the Ontology Alignment Evaluation Initiative." In: *Journal of Biomedical Semantics* 8 (2017), 56:1–56:28. DOI: 10.1186/s13326-017-0166-5

- Manel Achichi, Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Ian Harrow, Valentina Ivanova, Ernesto Jiménez-Ruiz, Elena Kuss, Patrick Lambrix, Henrik Leopold, Huanyu Li, Christian Meilicke, Stefano Montanelli, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Heiner Stuckenschmidt, Konstantin Todorov, Cássia Trojahn, and Ondrej Zamazal. "Results of the Ontology Alignment Evaluation Initiative 2016." In: *Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016).* Vol. 1766. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 73–129. URL: `http://ceur-ws.org/Vol-1766/oaei16_paper0.pdf`

- Manel Achichi, Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Ian Harrow, Valentina Ivanova, Ernesto Jiménez-Ruiz, Kristian Kolthoff, Elena Kuss, Patrick Lambrix, Henrik Leopold, Huanyu Li, Christian Meilicke, Majid Mohammadi, Stefano Montanelli, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Heiner Stuckenschmidt, Élodie Thiéblin, Konstantin Todorov, Cássia Trojahn, and Ondrej Zamazal. "Results of the Ontology Alignment Evaluation Initiative 2017." In: *Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017).* Vol. 2032. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 61–113. URL: `http://ceur-ws.org/Vol-2032/oaei17_paper0.pdf`

- Alsayed Algergawy, Michelle Cheatham, Daniel Faria, Alfio Ferrara, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Naouel Karam, Abderrahmane Khiat, Patrick Lambrix, Huanyu Li, Stefano Montanelli, Heiko Paulheim, Catia Pesquita, Tzanina Saveta, Daniela Schmidt, Pavel Shvaiko, Andrea Splendiani, Élodie Thiéblin, Cássia Trojahn, Jana Vatascinová, Ondrej Zamazal, and Lu Zhou. "Results of the Ontology Alignment Evaluation Initiative 2018." In: *Proceedings of the 13th International Workshop on Ontology Matching colocated with the 17th International Semantic Web Conference (ISWC 2018).* Vol. 2288. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 76–116. URL: `http://ceur-ws.org/Vol-2288/oaei18_paper0.pdf`

- Ernesto Jiménez-Ruiz, Tzanina Saveta, Ondvrej Zamazal, Sven Hertling, Michael R der, Irini Fundulaki, Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, Amina Annane, Zohra Bellahsene, Sadok Ben Yahia, Gayo Diallo, Daniel Faria, Marouen Kachroudi, Abderrahmane Khiat, Patrick Lambrix, Huanyu Li, Maximilian Mackeprang, Majid Mohammadi, Maciej Rybinski, Booma Sowkarthiga Balasubramani, and Cássia Trojahn. "Introducing the HOBBIT platform into the Ontology Alignment Evaluation Campaign." In: *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th*

International Semantic Web Conference (ISWC 2018). Vol. 2288. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 49–60. URL: http://ceur-ws.org/Vol-2288/om2018_LTpaper5.pdf

- Alsayed Algergawy, Daniel Faria, Alfio Ferrara, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Naouel Karam, Abderrahmane Khiat, Patrick Lambrix, Huanyu Li, Stefano Montanelli, Heiko Paulheim, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Élodie Thiéblin, Cássia Trojahn, Jana Vatascinová, Ondrej Zamazal, and Lu Zhou. "Results of the Ontology Alignment Evaluation Initiative 2019." In: *Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC 2019).* Vol. 2536. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 46–85. URL: http://ceur-ws.org/Vol-2536/oaei19_paper0.pdf

- Mina Abd Nikooie Pour, Alsayed Algergawy, Reihaneh Amini, Daniel Faria, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Clement Jonquet, Naouel Karam, Abderrahmane Khiat, Amir Laadhar, Patrick Lambrix, Huanyu Li, Ying Li, Pascal Hitzler, Heiko Paulheim, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Élodie Thiéblin, Cássia Trojahn, Jana Vatascinová, Beyza Yaman, Ondrej Zamazal, and Lu Zhou. "Results of the Ontology Alignment Evaluation Initiative 2020." In: *Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020).* Vol. 2788. CEUR Workshop Proceedings. CEUR-WS.org, pp. 92–138. URL: http://ceurws.org/Vol-2788/oaei20_paper0.pdf

- Mina Abd Nikooie Pour, Alsayed Algergawy, Florence Amardeilh, Reihaneh Amini, Omaima Fallatah, Daniel Faria, Irini Fundulaki, Ian Harrow, Sven Hertling, Pascal Hitzler, Martin Huschka, Liliana Ibanescu, Ernesto Jiménez-Ruiz, Naouel Karam, Amir Laadhar, Patrick Lambrix, Huanyu Li, Ying Li, Franck Michel, Engy Nasr, Heiko Paulheim, Catia Pesquita, Jan Portisch, Catherine Roussey, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Cássia Trojahn, Jana Vatascinová, Beyza Yaman, Ondrej Zamazal, and Lu Zhou. "Results of the Ontology Alignment Evaluation Initiative 2021." In: *Proceedings of the 16th In-*

ternational Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021). Vol. 3063. CEUR Workshop Proceedings. CEUR-WS.org, pp. 62–108. URL: `http://ceurws.org/Vol-3063/oaei21_paper0.pdf`

External Publications

Chapter 1 Introduction

"The Semantic Web is not 'merely' the tool for conducting individual tasks that we have discussed so far. In addition, if properly designed, the Semantic Web can assist the evolution of human knowledge as a whole."

Tim Berners-Lee, James Hendler and Ora Lassila

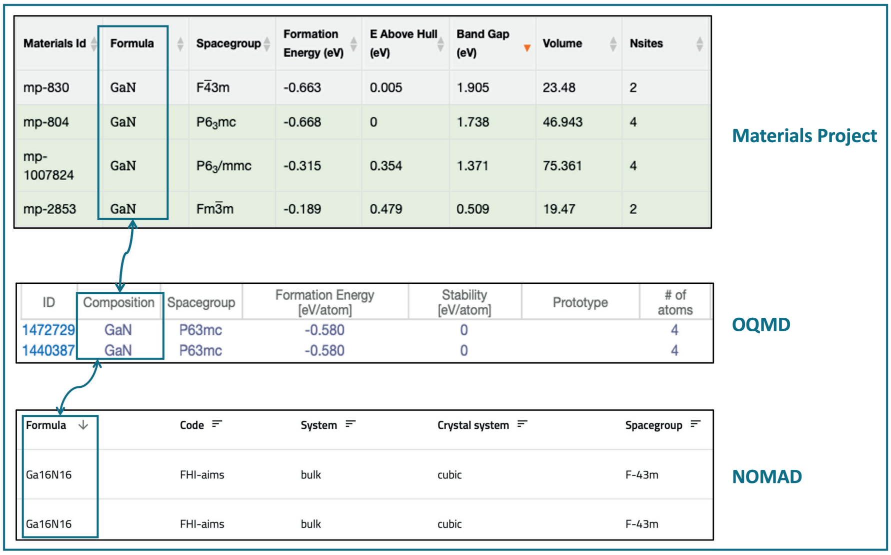
Tim Berners-Lee, James Hendler, and Ora Lassila proposed the idea of the Semantic Web, an extension of the Web, to enable exchange and reuse of data across applications [7]. The Semantic Web aims to make data on the Web machine-readable by introducing semantics to the data. The term *data* covers a wide variety of meanings including data models, schemas, vocabularies, as well as datasets and associated semantics [8]. Over the decades, a number of technologies have contributed to the layer cake of the Semantic Web. As the World Wide Web Consortium (W3C)[1] renders standards of Semantic Web technologies, some domains such as eScience and eBusiness are using Semantic Web-based technologies to assemble their domain knowledge and thus enhance their workflows [9].

[1]`https://www.w3.org/`

1.1 Motivation

This thesis is motivated by issues that relate to both the Semantic Web field as well as materials design, which is one of the sub-fields of the materials science domain. The materials science domain, like many other domains, is at an early stage when it comes to introducing Semantic Web-based technologies into its data-driven workflows. Over the last few decades, materials science has shifted towards its fourth paradigm, (big) data-driven science [10]. More and more materials scientists are recognizing the potential of data-driven techniques to accelerate the discovery and design of new materials. A large number of research groups and communities have thus developed a variety of data-driven workflows, including data repositories [1, 2] and data analytics tools. As data-driven techniques become more prevalent, more data is produced by computer programs and is available from various sources, which leads to challenges associated with reproducing, sharing, exchanging, and integrating data among these sources [11, 10, 12, 13, 14]. Figure 1.1 illustrates an example of searching for gallium nitride materials with the reduced chemical formula of GaN in three databases of the materials design domain, Materials Project [15, 16], OQMD (The Open Quantum Materials Database) [17, 18] and NOMAD (Novel Materials Discovery) [19, 20].

From the results, we can see that each of them contains a column that represents chemical composition, but with different column names or different insights (i.e., *'Formula'* for Materials Project and NOMAD, *'Composition'* for OQMD). The *'Formula'* column for Materials Project actually represents the reduced chemical formula. More detailed information regrading the chemical composition can be found based on the value of the *'Nsites'* column. For instance, for the second row of the result from Materials Project, we can derive that the unit cell formula is Ga_2N_2 based on the values of the *'Formula'* and *'Nsites'* columns. Meanwhile, the *'Formula'* column for NOMAD actually represents the unit cell formula rather than the reduced chemical formula. Unlike the other two databases, OQMD contains a column for reduced chemical formulas, but with a different column name (*'Composition'*). Such differences have to be addressed in order to integrate or exchange data from these data sources. Apart from such differences in terminology, the data that needs to be accessed or integrated from multiple data sources is typically heterogeneous in different models (i.e., relational data stored in relational databases, and hierarchical data stored in JSON data stores). Traditionally, ontology-based



Materials Id	Formula	Spacegroup	Formation Energy (eV)	E Above Hull (eV)	Band Gap (eV)	Volume	Nsites
mp-830	GaN	F$\bar{4}$3m	-0.663	0.005	1.905	23.48	2
mp-804	GaN	P6$_3$mc	-0.668	0	1.738	46.943	4
mp-1007824	GaN	P6$_3$/mmc	-0.315	0.354	1.371	75.361	4
mp-2853	GaN	Fm$\bar{3}$m	-0.189	0.479	0.509	19.47	2

ID	Composition	Spacegroup	Formation Energy [eV/atom]	Stability [eV/atom]	Prototype	# of atoms
1472729	GaN	P63mc	-0.580	0		4
1440387	GaN	P63mc	-0.580	0		4

Formula	Code	System	Crystal system	Spacegroup
Ga16N16	FHI-aims	bulk	cubic	F-43m
Ga16N16	FHI-aims	bulk	cubic	F-43m

Figure 1.1: An example of searching materials from Materials Project, OQMD and NOMAD.

data access and integration methods focus on data that follows relational data models. Therefore, it is challenging for a data integration system to manage requests to multiple different data sources (i.e., SQL queries to relational data sources or API (Application Programming Interface) requests to JSON or CSV data sources), and to provide integrated access to data from multiple data sources. Many other fields also face similar challenges. For instance, [21] discusses the problems of locating, retrieving, and integrating data in the biomedical field.

Moreover, these problems are very related to the more recently developed FAIR principles, which aim to make it easier for machines to locate and utilize data automatically, as well as for individuals to reuse data [22]. The FAIR principles state that data should be Findable, Accessible, Interoperable, and Reusable. Ontologies and ontology-based techniques are recognized as enablers of these principles. Using an ontology, knowledge of a domain can be represented in a shared and standardized way. By describing data using ontologies, the data will be more findable. By using ontologies for metadata representation, the level of accessibility can be raised. By using the same terminology as defined by ontologies, interoperability is enabled. Finally, since ontologies are shared and standardized, reusability is supported. However, developing ontologies is not an easy task. As a matter of knowledge representation, it is necessary to follow appropriate ontology engineering methodologies and gain a thorough understanding of the domain knowledge, which requires the participation of both ontology engineers and domain experts. Furthermore, we need to pay attention to maintaining ontologies throughout their life cycles.

That is to say, for one thing, we need an adapted ontology-driven data access and integration approach so that the heterogeneity of the underlying data can be addressed, as well as the diversity of ways in which data can be shared and queried. For another, we must have well-defined domain ontologies prior to implementing an ontology-driven approach to data access and integration.

An ontology-driven data access and integration approach can use GraphQL to orchestrate access to heterogeneous data sources. GraphQL [23] is a conceptual framework for building APIs for Web and mobile applications. It was publicly released in 2015 by Facebook, and the GraphQL ecosystem[2] has grown tremendously in terms of libraries[3] supporting different programming languages (such as JavaScript, Python, and Java), tools (such as Apollo[4] and GraphiQL[5]), and adopters (such as Airbnb, IBM, and Twitter). The framework introduces the notion of a GraphQL schema. The schema contains type definitions with fields, thereby describing the data that can be requested from the back-end data stores. The framework also contains a graph query language which allows to write GraphQL queries that ask for fields of objects. Besides the GraphQL schema and the query language, the implementation of a GraphQL server contains resolver functions for accessing back-end data sources and structuring data according to the GraphQL schema. However, although the GraphQL ecosystem is growing and GraphQL is used more and more, there is not much work on providing semantic and integrated access to multiple data sources, which is needed in many applications. GraphQL could be used to integrate data from different sources by building a GraphQL server over the existing data sources, in which the GraphQL schema provides a view over data from multiple sources. If a domain ontology can capture the semantics of data from multiple sources, we can make use of this ontology

[2]`https://landscape.graphql.org`
[3]`https://graphql.org/code/`
[4]`https://www.apollographql.com`
[5]`https://github.com/graphql/graphiql`

to guide the definition of the GraphQL schema to reflect concepts and relationships captured in an ontology. Meanwhile, semantic mappings, which are defined based on this ontology to describe how underlying data can be interpreted or annotated by the ontology, can be used in the resolver functions to provide information about how to access back-end sources and structure the obtained data according to the GraphQL schema. However, a semantics-aware approach to employing GraphQL for data integration does not exist. Furthermore, there are no formal methods for defining a GraphQL API. Therefore, developers have to implement the concrete details of a GraphQL server in terms of the schema and resolver functions manually. Among the contributions of this thesis is a formal method for automatically building a GraphQL server based on an ontology and semantic mappings.

We have seen that domain ontologies play an important role in representing domain knowledge and in facilitating the use of other Semantic Web-based technologies. In an ontology-driven approach to data access and integration, the coverage of the ontology may need to be enlarged when new databases are added or new kinds of data are added to existing databases. Therefore, it is vital that we maintain an ontology throughout its life cycle in order to make it more complete. However, developing and extending ontologies are not easy undertakings, and the results are not always complete. In addition to being problematic for modeling a domain accurately, such incomplete ontologies may also impact the quality of semantically enabled applications such as ontology-based search and data integration. Incomplete ontologies when used in semantically enabled applications can lead to valid conclusions being missed. For instance, in ontology-based search, queries are refined and expanded by moving up and down a hierarchy of concepts. Incomplete structure in ontologies influences the quality of the search results. In experiments in the biomedical field, an example was given where a search using the MeSH (Medical Subject Headings)[6] ontology in PubMed,[7] a large database with abstracts of research articles in the biomedical field, would miss 55 of the documents if the relation between the concepts *Scleral Disease* and *Scleritis* was missing [24]. Among the contributions of this dissertation is an approach for extending domain ontologies based on topic modeling, formal topical concept analysis and domain expert validation.

[6]`http://www.nlm.nih.gov/mesh/`
[7]`http://www.ncbi.nlm.nih.gov/pubmed/`

The work in this thesis is a part of a project in SeRC (*Swedish eScience Research Centre*), and is inspired by the work in the OPTIMADE consortium (*Open Databases Integration for Materials Design*). The project in SeRC has an aim of *Data-Driven Computational Materials Design.* More specifically, it aims to enhance the knowledge discovery process for materials design by using domain knowledge in the form of ontologies and Linked Data. The OPTIMADE consortium aims to make materials databases interoperable by developing a specification for a common REST API.

1.2 Problem formulation

The goal of this thesis is to offer a solution to the problem presented below:

> *How to provide semantics-aware data access and data integration over heterogeneous data, following different models, being shared and queried via different ways?*

Specifically, we have formulated this question in three parts:

- **RQ1**: How can the recently developed GraphQL be used for semantics-aware data access and data integration over heterogeneous data sources?

The first sub-question relates to how GraphQL can be used for data integration. One challenge highlighted in the previous section is that the heterogeneity over different data sources makes it difficult to access and integrate data, for ontology-based data access and integration approaches (e.g., [25], [26], [27]). To address this problem, we need to facilitate the usage of ontologies in a situation where heterogeneity exists. With regards to this research question, we pursue the following objective: to design an ontology-driven data access and integration framework in which a GraphQL server plays a role in accessing underlying data sources by providing an (integrated) view of the data.

- **RQ2**: How can ontologies be leveraged to generate GraphQL APIs for semantics-aware data access and data integration?

The second sub-question relates to how a GraphQL server can be generated automatically to avoid constructing the GraphQL server from scratch. A problem when applying GraphQL for data integration is that there are no existing formal methods for defining a GraphQL API aiming at data integration. With regards to this research question, we pursue the following objectives: to design a formal method to generate a GraphQL schema based on an ontology and a generic implementation of resolver functions based on semantic mappings; to evaluate the framework with experiments over a synthetic benchmark dataset, as well as a dataset from the materials design field; and to construct a domain ontology for the materials design field prior to evaluating and applying the framework in the field.

- **RQ3**: How can domain ontologies be extended by mining unstructured text, with validation from domain experts?

The third sub-question relates to extension of domain ontologies. To answer this research question, we pursue the following objective: to design an approach for extending domain ontologies based on topic modeling, formal topical concept analysis and domain expert validation; and to apply this approach in the materials science field.

1.3 Contributions

With a high-level GraphQL-based framework for data access and integration and five contribution components related to different parts of the framework, this thesis contributes in three respects to address the three research questions. We show them as follows:

- To answer **RQ1**, we outline a GraphQL-based data access and integration framework in which an ontology drives the generation of the GraphQL server.
- To answer **RQ2**, one contribution is that we implement a prototype of the framework in terms of ontology-based GraphQL server generation (OBG-gen) (**C1**). We evaluate our approaches by conducting experiments over a synthetic benchmark dataset and also over a dataset collected from the materials design field. For the evaluation in the materials design domain we make another contribution, which is the Materials Design Ontology (MDO) (**C2**). MDO demonstrates the ability to increase interoperability among different materials databases and has attracted the interest of database providers. After that, we show the application of our approaches, in terms of MDO and the GraphQL-based framework, in OPTIMADE (**C3**).

- Within the scope and vision of the framework, and to answer **RQ3**, we propose an approach for ontology extension based on phrase-based topic modeling, formal topical concept analysis, and domain expert validation (**C4**). We conduct experiments on the approach over the nanotechnology domain and the materials design domain. Based on the results of the experiments, we evaluate our approach, and produce valuable candidates (**C5**) that can be used to extend relevant domain ontologies.

1.4 Research methods

In accordance with the formulated problems and relevant objectives described in the previous section, this dissertation intends to address issues in and contribute to both the Semantic Web field and the materials design field. We have employed several scientific research methods in our research.

Our first step was to conduct systematic literature reviews on relevant topics in both the Semantic Web field and the materials design field in order to assess the current state of the art. In particular, the topics comprise data management, databases, and ontologies, with focuses on materials science, ontology extension, ontology-based data access and integration, as well as GraphQL. The systematic literature review aims to identify any gaps in current research, to summarize the existing evidence of a treatment or technology, and to provide a framework or background for positioning new research activities [28]. Based on systematic literature reviews, we were able to identify the challenges related to data access and integration, specific problems that need to be resolved and hypotheses that underlie our research. The hypotheses of our work are shown below:

- **Hypothesis 1:** The recently developed GraphQL can be used to assemble an integrated view of underlying data and manage requests to underlying data sources in an ontology-driven data access and integration scenario.
  - GraphQL servers can be automatically generated based on proper domain ontologies and semantic mappings, in order to reduce the need to construct GraphQL servers from scratch.
- **Hypothesis 2:** Ontologies and ontology-based techniques can help in making data FAIR for the materials science domain.

– In one respect, we require domain ontologies with an emphasis on describing semantics in order for data integration and access to be possible. In another respect, we need approaches that can generate candidates for extending existing domain ontologies.

In the second step, we proposed specific conceptual frameworks while answering the research questions. By building conceptual frameworks, researchers can obtain a better understanding of the core concepts of the study and find the relationships among these concepts [29, 30]. Then, we applied the prototyping methodology to develop our systems incrementally based on the conceptual frameworks. The prototyping and incremental development allow us to implement a partial system or a working version of the system which can be reviewed and further improved. During the development, we maintained the deliverables via GitHub repositories.[8], [9] Finally, both qualitative and quantitative evaluations were conducted, and an application in the materials design field was enabled. We considered quantitative factors, such as query execution time when we evaluate our GraphQL-based framework for data access and integration, and precision when we evaluate our approach for ontology extension. We took the quality criteria such as generalizability into account during the evaluation by conducting experiments on our GraphQL-based data access and integration framework using a synthetic benchmark dataset. Generalizability refers to whether or not the results generated in one study can be applied or extended to wider groups or different users and situations [31, p. 280]. Additionally, in terms of ontology development, we followed some ontology engineering methodologies and best practices to develop a domain ontology for the materials design field. We maintained the deliverables via a GitHub repository.[10]

1.5 Thesis outline

The outline of this thesis and the mappings among chapters, research questions, contributions are depicted in Figure 1.2.

We introduce concepts related to ontologies, RDF, SPARQL and data integration in **Chapter 2**, as well as the background of the materials design

[8] `https://github.com/LiUSemWeb/OBG-gen`
[9] `https://github.com/LiUSemWeb/ToPMine-FTCA`
[10] `https://github.com/LiUSemWeb/MDO`

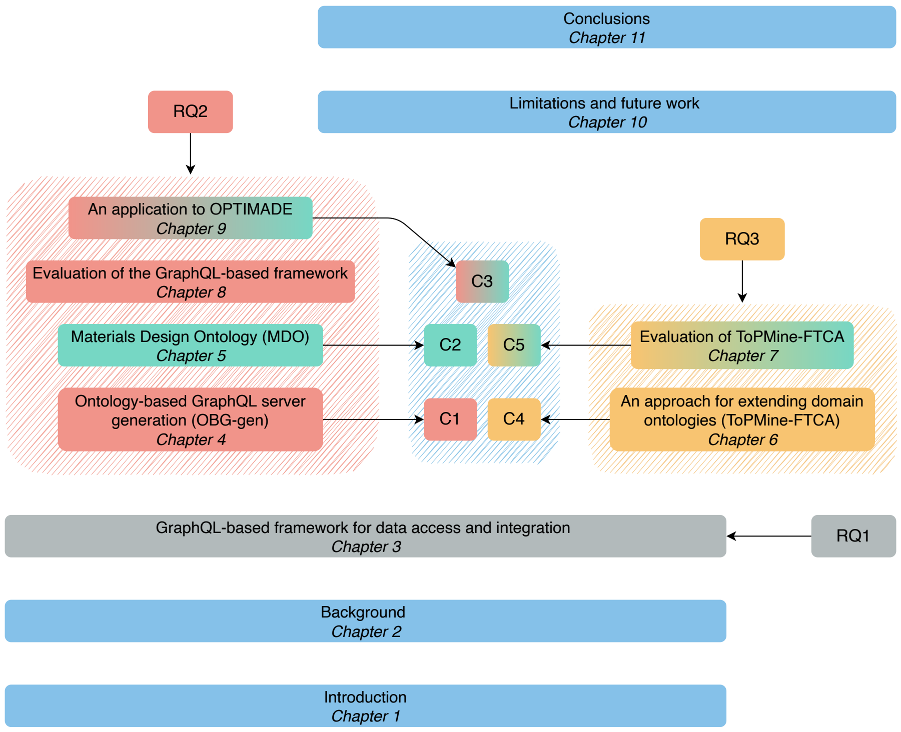

Conclusions
Chapter 11
Limitations and future work
Chapter 10
RQ2
An application to OPTIMADE
Chapter 9
Evaluation of the GraphQL-based framework
Chapter 8
Materials Design Ontology (MDO)
Chapter 5
Ontology-based GraphQL server generation (OBG-gen)
Chapter 4
C3
C2
C5
C1
C4
RQ3
Evaluation of ToPMine-FTCA
Chapter 7
An approach for extending domain ontologies (ToPMine-FTCA)
Chapter 6
GraphQL-based framework for data access and integration
Chapter 3
RQ1
Background
Chapter 2
Introduction
Chapter 1

Figure 1.2: Mappings among thesis chapters and research questions, contributions.

domain and FAIR data principles. In **Chapter 3**, we outline the GraphQL-based framework for data access and integration. One important component of this framework is the ontology-based GraphQL server generation of which we present the implementation in **Chapter 4**. The implementation contains the GraphQL schema generation based on an ontology and a generic implementation of resolver functions based on semantic mappings. We present formal methods for automatically generating a GraphQL server in terms of the GraphQL schema and a generic resolver function.

Within the scope and vision of the framework presented in Chapter 3, we turn our focus to another essential component of the framework that relates to ontology engineering. In **Chapter 5**, we present the Materials Design Ontology, which is a domain ontology for the materials design field and is developed by us with the purpose of making data over multiple materials databases FAIR. Ontologies and databases relevant to materials design are also discussed. In **Chapter 6**, we present an approach for ontology extension based on topic modeling, formal topical concept analysis, and domain expert validation. In **Chapter 7**, we evaluate this approach by conducting experiments in the nanotechnology domain and the materials design domain. In **Chapter 8** we turn our attention to evaluating the framework presented in Chapter 3. In **Chapter 9**, we introduce the usage of MDO and the GraphQL-based framework for data access and integration to OPTIMADE. In **Chapter 10** we discuss the limitations of our work and show some interesting directions for future work. Towards the end of the thesis, the research questions and contributions are reviewed in **Chapter 11**.

Chapter 2

Background

In this chapter, we provide an overview introduction to areas that are pertinent to this thesis. As a first step, we introduce ontologies in Section 2.1 from the perspective of knowledge representation, as well as RDF and SPARQL. In Section 2.2, we present the background of data integration with a focus on ontology-based data access and integration. Since materials design is an application domain to which this thesis intends to make a contribution, we then introduce the materials design field in Section 2.3. In Section 2.4, we provide an introduction to FAIR principles. As a final step, we provide a summary in Section 2.5.

2.1 Ontologies, RDF, SPARQL

Ontologies. The term *ontology* originates in philosophy, in which it is the science of what is, of the kinds and structures of objects, properties, and relationships in every area of reality [32, 33]. It is since 1980, when Alexander et al. [34] presented the technique known as *"ontological analysis"* from a knowledge engineering perspective that ontologies were introduced into many communities in computer science [33]. Ontologies can be viewed, intuitively, as defining the terms, relations, and rules that combine these terms and relations in a domain of interest [35]. Through ontologies, people and organizations are able to communicate by establishing a common terminology. They provide the basis for interoperability between systems and are applicable as an index to a repository of information as well as a query model and a navigation model for data sources. Moreover, they are often used as a foundation for integrating data sources, thereby alleviating the heterogeneity issue. The benefits of using ontologies are their improved reusability, share-ability and portability across platforms, as well as their increased maintainability and reliability. On the whole, ontologies allow a field to be better understood and allow information in that field to be handled much more effectively and efficiently (e.g., knowledge representation for bioinformatics discussed in [36]).

From a knowledge representation point of view, ontologies usually contain four components: (i) concepts that represent sets or classes of entities in a domain, (ii) instances that represent the actual entities, (iii) relations, and (iv) axioms that represent facts that are always true in the topic area of the ontology. Relations can represent relationships among concepts. Axioms can illustrate domain restrictions, cardinality restrictions, or disjointness restrictions. Depending on the components and information related to the components they contain, ontologies can be classified. As an example, Figure 2.1 represents a small piece of the Materials Design Ontology (MDO) regarding some core concepts and relationships (more details of MDO are given in Chapter 5). The open-headed arrows represent axioms that represent is-a relationships that is, if A is a B, then all entities belonging to concept A also belong to concept B. We say that A is a sub-concept of B. In this example

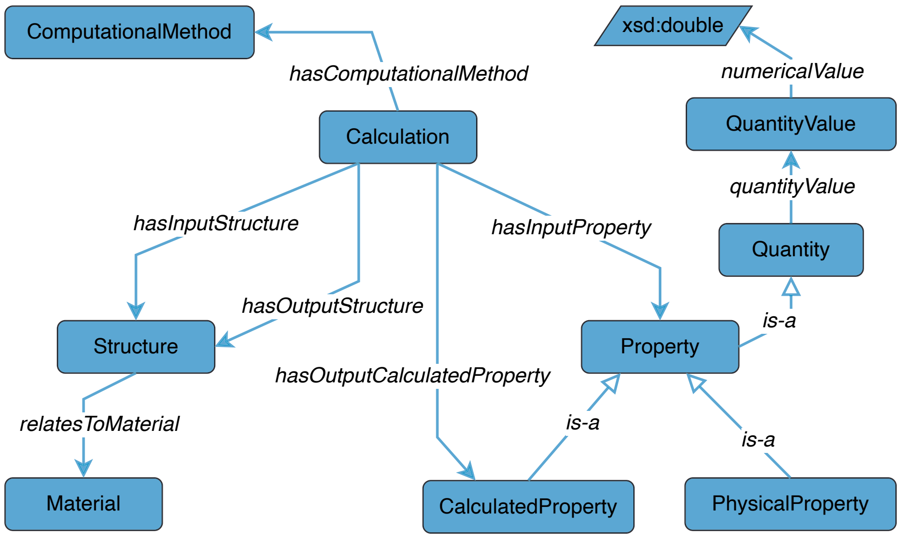


Figure 2.1: An outline of Materials Design Ontology.

we have it that *CalculatedProperty* and *PhysicalProperty* are sub-concepts of *Property*, which is a sub-concept of *Quantity*. Therefore, all *CalculatedProperty* and *PhysicalProperty* entities are *Property* entities which are *Quantity* entities. The is-a relation is transitive such that, for instance, a *CalculatedProperty* entity is also a *Quantity* entity. The closed-headed arrows represent general relations among concepts other than is-a relations. For instance, the *Calculation* concept has a connection to the *CalculatedProperty* concept represented by the *hasOutputCalculatedProperty* relation. Additionally, a relation can exist between a concept and a data type reference. For instance, *QuantityValue* has a connection to the data type reference *xsd:double* represented by the *numericalValue* relation. This means that each entity of the *QuantityValue* concept can be associated with a double type value by having a *numericalValue* connection.

In Figure 2.2 we show the part of MDO represented using the ontology development system Protégé.[1] On the left hand side the concepts and the is-a hierarchy are shown. The is-a relations are represented by indentation. For instance, *CalculatedProperty* is a sub-concept of *Property*, which in turn is a sub-concept of *Quantity*. On the right-hand side of Figure 2.2 information related to the axioms of *Structure* are shown using a special notation reflecting constructs in the representation language OWL (Web Ontology Language),[2, 3] a knowledge representation language that is often used for representing ontologies and that is based on description logics [37]. Description logics are a family of knowledge representation languages that include formalizations. There are three basic building blocks of such a language, namely: (i) atomic

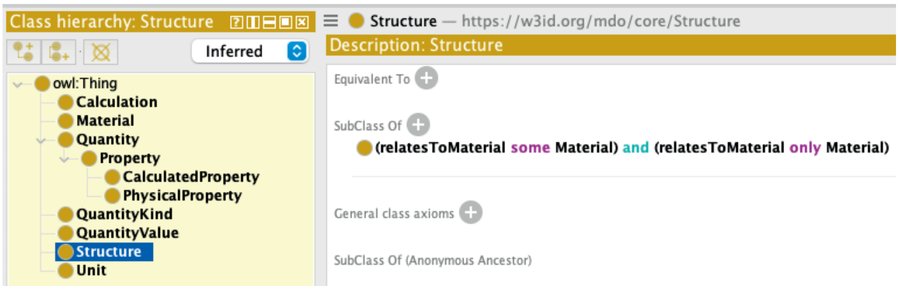


Figure 2.2: Materials Design Ontology opened in Protégé.

[1] `https://protege.stanford.edu/`
[2] `http://www.w3.org/TR/owl-features/`
[3] `http://www.w3.org/TR/owl2-overview/`

concepts (unary predicates) such as *Calculation* and *Structure*, (ii) atomic roles (binary predicates) such as *relatesToMaterial*, and (iii) individuals (constants) [37]. On the basis of these basic building blocks and logical constructors such as conjunction ($\sqcap$), disjunction ($\sqcup$), universal restriction ($\forall$), existential restriction ($\exists$), and general concept inclusion ($\sqsubseteq$), we can represent more complex concepts or semantics. In Figure 2.2, the *Structure* concept contains a definition, which can be represented in a description logic language as $Structure \sqsubseteq \exists relatesToMaterial.Material \sqcap \forall relatesToMaterial.Material$. This means that a *Structure* entity is a sub-concept of an entity that may have a *relatesToMaterial* relation, and the range of this relation must be a *Material* entity.

RDF. The Resource Description Framework (RDF) is recommended by the W3C [38], and can be used for representing graph data and supporting data exchange. The core structure of the RDF-based data model is a set of triples where each triple has a *subject*, a *predicate* and an *object* [38]. A set of such triples is called an RDF graph in which each node represents a subject or an object and each edge represents a predicate [38]. In an RDF graph, IRIs (Internationalized Resource Identifiers) are used to represent globally unique identifiers for resources. The Internationalized Resource Identifier is an internet protocol standard which extends the Uniform Resource Identifier (URI) protocol by permitting more Unicode characters [38]. In an RDF graph, a subject can be an IRI, or a blank node which is an anonymous resource; a predicate is an IRI; an object can be an IRI, a literal or a blank node. For a more detailed introduction to RDF, we refer the reader to [38]. Listing 2.1 illustrates an example RDF graph representing data from the materials design domain. At the beginning of the example, we have several namespace definitions which are used for abbreviated URIs (line 1 to line 3). After that, as we can see, there are four triples in total. The first two triples have the same subject, which is defined using the IRI *http://example.org/materials-design/calculation_1*. The last two triples have the same subject, *http://example.org/materials-design/property_1*. The predicate `rdf:type` is used to classify a resource as an instance of a concept. In our example, the two kinds of subjects represent resources are instances of `core:Calculation` and `core:CalculatedProperty`, respectively. These two concepts are from MDO. We also have predicates defined as `core:hasOutputCalculatedProperty` and `core:PropertyName` to represent relationships between resources. The object of the last triple is a literal which is a string (“Band Gap”).

Listing 2.1: An example RDF graph.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX core: <https://w3id.org/mdo/core/> .
PREFIX ex: <http://example.org/materials-design/> .

ex:calculation_1 rdf:type core:Calculation .
ex:calculation_1 core:hasOutputCalculatedProperty ex:property_1 .
ex:property_1 rdf:type core:CalculatedProperty .
ex:property_1 core:PropertyName "Band Gap" .
```

SPARQL. SPARQL is the W3C recommendation for querying RDF graphs [39]. SPARQL enables users for querying data that can be mapped to RDF. We refer to the syntax definition of SPARQL in [40]. This work presents the definition of the SPARQL graph patterns recursively as below:

- A tuple from $(I \cup V) \times (I \cup V) \times (I \cup L \cup V)$ is a graph pattern, where I is a set of IRIs, L is a set of literals and V is an infinite set of variables disjoint from I and L. A graph pattern is called a triple pattern if there is just one single tuple.

- If P_1 and P_2 are graph patterns, then expressions (P_1 AND P_2), (P_1 OPT P_2), and (P_1 UNION P_2) are also graph patterns. They are called a *conjunction graph pattern*, an *optional graph pattern* and a *union graph pattern*, respectively.

- If P is a graph pattern, and R is a SPARQL built-in condition, then expression (P FILTER R) is a graph pattern, which is also called a *filter graph pattern*.

Listing 2.2 illustrates an example SPARQL query over the data represented in the RDF graph in Listing 2.1. This query retrieves all the properties and the corresponding property names. From line 5 to line 8, we have the *WHERE* clause which specifies the graph pattern to be matched. The *SELECT* clause (at line 4) specifies the variables to be projected from the graph pattern.

Listing 2.2: An example SPARQL query.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX core: <https://w3id.org/mdo/core/> .

SELECT ?property ?property_name
WHERE {
  ?property rdf:type core:Property;
            core:PropertyName ?property_name.
}
```

2.2 Data integration

Data integration is regarding combining data that resides at multiple different sources [41, 42, 43]. Ideally, a data integration system should enable unified access to a number of data sources [41, 43]. Formally, according to [41], a data integration system can be formalized as a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{G}$ is the *global schema*, expressed in a language $\mathcal{L}_\mathcal{G}$ over an alphabet $\mathcal{A}_\mathcal{G}$;
- $\mathcal{S}$ is the *source schema*, expressed in a language $\mathcal{L}_\mathcal{S}$ over an alphabet $\mathcal{A}_\mathcal{S}$;
- $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$, constituted by a set of *assertions* that define mappings from queries over the source schema $\mathcal{S}$ to queries over the global schema $\mathcal{G}$ (similarly for mappings from queries over $\mathcal{G}$ to queries over $\mathcal{S}$). Such a mapping specifies correspondences between concepts in the global schema and those in the source schema.

Ontology-based data integration (OBDI) is a form of data integration in which an ontology plays the role of a global schema that captures domain knowledge [44]. Usually, in an information system with only one single data source, the formal treatment of OBDI is identical to that of ontology-based data access (OBDA) [44, 45]. In this thesis, we generally refer to both OBDI and OBDA as OBDA. OBDA, as a semantic technology, aims to facilitate access to different underlying data sources [46]. Traditionally, these underlying data sources are considered to be relational databases. Ontologies play the role of global views over multiple data sources. There are different ways to implement an OBDA system. Generally, these systems can be categorized into two types, namely, data warehouse-based approaches and virtual approaches. These two categories of methods both make use of semantic mappings in order to overcome the differences between ontologies and local schemas, but in different ways [47, 48]. In a data warehouse-based approach, data from multiple sources are usually loaded or stored in a centralized storage, which is the *warehouse* [49, 43], based on semantic mappings. We refer to the data in such warehouses as materialized data. Depending on the aims or functionalities of a system, the materialized data could be stored in local databases or transformed into RDF graphs. Therefore, queries are evaluated against the materialized data. In a virtual approach, data is retained at the original sources and *mediators* are used to translate queries defined in terms of a global or mediated schema into queries defined in terms of each data source's local schema, based on semantic mappings. Therefore, queries are evaluated and executed against each data source. SPARQL queries are widely supported by data integration systems that use ontologies as global schemas.

A number of semantic mapping definition languages have been proposed over the years. R2RML (RDB to RDF Mapping Language) is a language, one of the two recommendations by the RDB2RDF W3C Working Group,[4] to define semantic mappings [50]. R2RML supports transformation rules defined by users, while the other recommendation, Direct Mapping [51], does not. Another language is RDF Mapping Language (RML) [52, 53], which allows underlying data in formats beyond relational databases and is a superset of R2RML. RML can also deal with data from CSV, JSON, and XML data sources. In Section 4.2.2 of Chapter 4 we introduce more details of RML, of which we make use in our work.

2.3 Materials design domain

The design of materials is a technological process that has many applications. Most often, the goal is to achieve a set of desired material properties for an application within certain limitations, such as avoiding or eliminating toxic or critical raw materials. Such raw materials are of strategic and economic importance for the economy but have a high risk associated with their supply [54]. The development of condensed matter theory and materials modeling has made it possible to achieve quantum mechanics-based simulations that can generate reliable materials data by using computer programs [55]. Over the years, quite a number of materials databases have emerged. A common use of these databases is to find materials with desirable properties as shown in the

[4]`https://www.w3.org/2001/sw/rdb2rdf/`

data-driven materials design example discussed in [56]. At the same time, several global efforts are underway to assemble and curate databases combining experimentally measured and computationally predicted properties of materials, and also to make them interoperable. For instance, the *Open Databases Integration for Materials Design* (OPTIMADE)[5] consortium aims to make materials databases interoperable by developing a specification for a common REST API (Application Programming Interface). Some of the work in this thesis is inspired by the work in the OPTIMADE consortium and makes an application to OPTIMADE. We introduce more details of OPTIMADE in Section 5.1.4 of Chapter 5, and we discuss the application in Chapter 9.

As databases in the materials design domain are heterogeneous in nature and data is usually shared via APIs such as Web APIs in the domain, there are a number of challenges to using them in an integrated way in the materials design workflow. For instance, retrieving data from more than one database means that users have to understand and use different APIs or even different data models to reach an agreement. APIs providing connections or communications between computer applications or among components of a software [57, 58], have been widely used, not only for exposing functionalities but also for sharing data [58, 59]. Although APIs can establish guidelines regarding how to access data held in a specific database, integrating data that is accessed via APIs is a challenging problem for both the materials science field and the Semantic Web field. Data obtained via API requests is not usually explicitly grounded in semantics [60]. The underlying data models are usually obfuscated by APIs.

2.4 FAIR data principles

The FAIR principles were defined in 2016 by a wide range of scientists and organizations representing academia, industry, funding agencies, and scholarly publishers [22]. The principles state that data should be Findable, Accessible, Interoperable, and Reusable, respectively, with a goal of allowing machines to automatically find and use data, and allowing individuals to reuse the data [22]. Findable refers to the fact that data should be easy to find, accessible to the fact that it should be clear how to access the data, interoperable to the fact that the data needs to be integrated with other data and be usable

[5] `https://www.optimade.org/`

by applications and workflows, and reusable to the fact that data should be well described such that the data can be replicated or combined in different settings.[6] One way to make data FAIR is to annotate or classify data by using ontologies. Ontologies can yield the annotations of data and the mappings between data and concepts, relationships, which means that we can append semantics to underlying data. From an application point of view, a general data access or integration framework capable of providing a unified view of data from multiple data sources, managing requests to these data sources and responding explicitly to users with semantics, can increase the data interoperability.

As we mention at the very beginning of Chapter 1, the term *data* covers a wide variety of meanings, which means it can also represent metadata such as vocabularies and ontologies used to annotate and interpret the data. It is also important that we make such metadata FAIR. To make a vocabulary FAIR, some rules have been identified in [61]. For instance, registering vocabularies in open repositories such as Linked Open Vocabularies (LOV)[7] can enable findability; making relevant URIs resolve can enable accessibility such as reserving secure and permanent URLs (Uniform Resource Locators) from the W3C Permanent Identifier Community Group[8]; creating vocabularies with standard means such as SKOS (Simple Knowledge Organization System)[9] or OWL can enable interoperability; and adding rich metadata to data can enable reusability. Additionally, there are a number of guidelines designed to make ontologies FAIR [62]. For instance, metadata registries and annotations can help in findability; URI design and content negotiation can help with accessibility; serving ontologies in different standard serializations can help with interoperability; and metadata description and diagram guidelines can help with reusability.

2.5 Summary

In this chapter, we have introduced ontologies, RDF, SPARQL and data integration with a focus on ontology-based approaches. Following that, we moved on to the materials design domain, and then introduced FAIR data principles. The work covered in this thesis is particularly relevant to these topics. On the

[6]`https://www.go-fair.org/fair-principles/`
[7]`https://lov.linkeddata.es/dataset/lov/`
[8]`https://www.w3.org/community/perma-id/`
[9]`https://www.w3.org/2004/02/skos/`

basis of this background knowledge, in the following chapters we elaborate on how this thesis addresses the research questions and describe the contributions of this thesis.

Chapter

3

GraphQL-based framework for data access and integration

In this chapter, we present a GraphQL-based framework for data access and integration in which the GraphQL server is generated automatically based on an ontology and semantic mappings. First, for the sake of background knowledge, we introduce GraphQL in Section 3.1. We then introduce the outline of the framework in Section 3.2. The chapter ends with a summary in Section 3.3.

3.1 GraphQL

GraphQL schemas and GraphQL resolver functions are basic building blocks in the implementations of GraphQL servers. The former describe how users can retrieve data using GraphQL APIs. The latter contain program code including how to access data sources and structure the obtained data according to the schema. We introduce GraphQL schemas and GraphQL resolver functions in Section 3.1.1 and Section 3.1.2, respectively.

3.1.1 GraphQL schemas

In a GraphQL API, the GraphQL schema defines types, their fields, and the value types of the fields. Such a schema represents a form of vocabulary supported by a GraphQL API rather than specifying what the data instances of an underlying data source may look like and what constraints have to be guaranteed [63]. There are six different type definitions in GraphQL, which are scalar type, object type, interface type, union type, enum type and input object type. Figure 3.1 depicts a GraphQL schema example.

An object type represents a list of fields and each field has a value of a specific type such as object type or scalar type. A scalar is used to represent a value such as a string. In Figure 3.1, there are three basic object type definitions, which are `University`, `Department`, and `Professor`. They all have field definitions which represent the relationships to scalar types or to other object types. For instance, the `University` type has a field definition `UniversityID` of which the value type is `String`, and a field definition `departments` of which the value type is a list of `Departments`. GraphQL allows defining abstract types by supporting the interface type and the union type. An interface type defines a list of fields and allows object types to implement. An object type can then implement an interface type with the requirement that the object type includes all fields defined by the interface type. The schema in Figure 3.1 contains an interface type, `Author` with an `AuthorID` field of which the value type is `String`. The object type `Professor` implements `Author` and must have the same definition for `AuthorID` field as that in `Author`. A union type defines a list of possible types. An enum type describes the set of possible values that are in scalars. For more details of union types and enum types, we refer the reader to the latest GraphQL specification in [23].

GraphQL allows fields to accept arguments to configure their behavior [23]. These arguments can be defined by input object types. An input object type defines an input object with a set of input fields; the input fields are either scalars, enums, or other input objects. This allows arguments to accept arbitrarily complex structs, which can capture notions of filtering conditions. For instance, according to the definitions of `UniversityFilter` and `StringFilter`, we can define an input argument as `UniversityID:{_eq:"u1"}` to capture the meaning of "*UniversityID is equal to 'u1'*", where `_eq` represents the equal to operator. In our implementation presented in Chapter 4, `_and`, `_or` and `_not` are used to represent boolean expressions. For instance, `_or:[{UniversityID:{_eq:"u1"}}, {UniversityID:{_eq:"u2"}}]` represents the expression "*UniversityID is equal to 'u1' or 'u2'*". In the example schema, we use the term `filter` to represent the name of an input argument. This is just an informal way to state input arguments representing filter conditions. Such input arguments defined as input objects are not built-in constructs of GraphQL. Therefore, their meanings are essentially defined by the program code of the GraphQL

```
type University{
  UniversityID: String
  departments: [Department]
}
type Department{
  DepartmentID: String
  head: String
}
interface Author{
  AuthorID: String
}
type Professor implements Author{
  AuthorID: String
  doctoralDegreeFrom: [University]
}
input UniversityFilter{
  UniversityID: StringFilter
  departments: DepartmentFilter
  _and: [UniversityFilter]
  _or: [UniversityFilter]
  _not: UniversityFilter
}
input DepartmentFilter{
  DepartmentID: StringFilter
  head: StringFilter
  _and: [DepartmentFilter]
  _or: [DepartmentFilter]
  _not: DepartmentFilter
}
input StringFilter{
  _eq: String
  _in: [String]
  _neq: String
  _nin: [String]
  _like: String
}
type Query{
  UniversityList(filter: UniversityFilter): [University]
  DepartmentList(filter: DepartmentFilter): [Department]
  AuthorList: [Author]
  ProfessorList: [Professor]
}
```

Figure 3.1: A GraphQL schema example.

server implementation, i.e., the resolver functions which manage requests to underlying data sources and structure the returned data according to the GraphQL schema.

Additionally, a GraphQL schema supports defining types that represent operations such as query and mutation. The schema presumes the `Query` type as the query root operation type. As Figure 3.1 shows, in the `Query` type definition, there are four field definitions, which are `UniversityList`, `DepartmentList`, `AuthorList`, and `ProfessorList`. For instance, the returned type of `UniversityList` is `[University]`, a list of `Universities`. The `UniversityList` takes an argument defined as `UniversityFilter` as an input for capturing the notion of a filtering condition.

3.1.2 GraphQL resolver functions

In a GraphQL API, apart from the GraphQL schema defining types, their fields, and the value types of the fields, resolver functions are responsible for populating the data for fields of types in the GraphQL schema. For instance, for the schema example shown in Figure 3.1, there are four fields defined in the `Query` type. Therefore, in the GraphQL server implementation, we are supposed to define resolver functions to populate data for these fields, `UniversityList`, `DepartmentList`, `AuthorList`, and `ProfessorList`. In our implementation presented in Chapter 4, we assume that the GraphQL schema supports a query that retrieves all the instances for each interface type or object type. Therefore, we use the name of each interface type or object type concatenated with 'List' as the name of a field in the `Query` type, where the returned type is a list of the interface or object type. This is just an informal way to state the behavior of a field in the `Query` type. To emphasize, what a GraphQL query can retrieve over the underlying data sources relies on how the resolver function is implemented. For instance, if the underlying data source is a relational database, the resolver function should contain code specifying the SQL query to be evaluated.

Listing 3.1 illustrates an example resolver function (written in JavaScript syntax) for the `UniversityList` field. We assume that the underlying data source is a relational database that contains a table named *university* with a column named *id*. In line 2 and line 3, given an input argument representing the id of a university (*university_id*), a query is evaluated against the relational database. In line 4, the data is structured according to the *University* object defined in the JavaScript code which corresponds to the `University` type definition in the schema shown in Figure 3.1.

Listing 3.1: An example resolver function for the `UniversityList` field.

```javascript
const UniversityList = (university_id) => {
  let data = db_conection.select().from('university')
                .where('id', university_id);
  let allUniversities = data.then(rows => new University(rows[0]));
  return allUniversities;
};
```

3.2 Overview of the framework

Figure 3.2 illustrates the framework for data access and integration based on GraphQL in which an ontology drives the generation of GraphQL server that provides integrated access to data from heterogeneous data sources. These data sources may be based on different schemas and formats and may be accessed in different ways (e.g., as tabular data accessed via SQL queries or as JSON-formatted data accessed via API requests). To address the heterogeneity, the framework relies on an ontology that provides an integrated view of the data from the different sources, and corresponding semantic mappings that define how the data from the underlying data sources is interpreted or annotated by the ontology (arrows **(a)**) and **(b)**). Furthermore, two processes

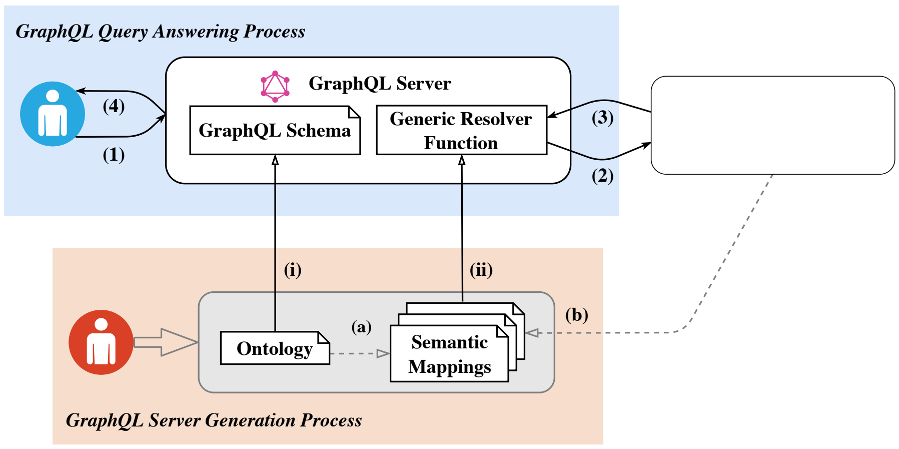

GraphQL Query Answering Process
GraphQL Server
GraphQL Schema
Generic Resolver Function
(4)
(1)
(3)
(2)
(i)
(ii)
Ontology
(a)
Semantic Mappings
(b)
GraphQL Server Generation Process

Figure 3.2: GraphQL-based framework for data access and integration.

are defined. The first process generates the GraphQL server. The second process deals with answering queries and is performed after the GraphQL server is set up. In accordance with these two processes, we have two types of intended users or developers in the framework. One type is users or developers of the GraphQL server generator, who should have prior knowledge of the ontology, semantic mappings and the domain. The other type is end users using a GraphQL server for data access and integration, who may or may not be familiar with the Semantic Web or ontologies. For the purpose of writing GraphQL queries, they need basic prior knowledge of GraphQL, which can be learned from the self-documenting API of the generated GraphQL server showing the schema. We introduce more details about these two processes in Section 3.2.1 and Section 3.2.2, respectively.

3.2.1 GraphQL server generation process

This process includes generating both a GraphQL schema for the API provided by the server (arrow `(i)`) and a generic resolver function (arrow `(ii)`). Given an ontology as an integrated view of data from multiple data sources, we propose a method for generating a GraphQL schema based on this ontology, with the result that the schema becomes a view of the data to be integrated. Additionally, we propose a generic implementation of resolver functions that takes semantic mappings as inputs, so that the server is able to get data from underlying data sources. In Chapter 4, we elaborate on the implementation of our approaches for generating the GraphQL schema and the generic resolver function. This GraphQL server generation process does not need to be repeated unless the ontology or the semantic mappings change. After this generation process, the GraphQL server can be set up.

In this GraphQL server generation process, we require users or developers who are familiar with the query mechanisms of underlying data sources, domain ontologies that can be used for data access or integration. Consequently, they can define the scope of the ontology that will be used for generating the GraphQL schema for the server, as well as the semantic mappings that will be used for generating the generic resolver function. This type of automatic generation of GraphQL servers based on ontologies and semantic mappings can also benefit general GraphQL application developers, since it can eliminate the need to build GraphQL servers from scratch.

3.2.2 GraphQL query answering process

During this process the query is validated against the GraphQL schema (arrow `(1)`); the underlying data sources are accessed via resolver functions, the retrieved data is combined, the data is structured according to the schema (arrows `(2)` and `(3)`); and finally the query result is returned (arrow `(4)`).

A GraphQL query example and corresponding query result are shown in Figure 3.3. The example query is: "*Get the university including the head of each department where the UniversityID is 'u1'*". The query takes as an input an argument defined as `filter: {UniversityID:{_eq:"u1"}}`, which follows the syntax of the input object type `UniversityFilter`. As we mention in Section 3.1.1, the meaning of an input argument defined as an input object type is essentially determined by the program code of the resolver functions. Thus the query example shown in Figure 3.3a illustrates one way that we make use of input objects to represent filtering conditions. In general, however, the input object types can be used in various ways for any field, depending on the implementation of the GraphQL server.

It has been noted that domain users are the intended users of GraphQL servers, regardless of whether they have prior knowledge of the Semantic Web or ontologies. In order to write GraphQL queries, they only need to have a basic understanding of GraphQL, which can easily be explored via the GraphQL API provided by the server.

```
{
  UniversityList(
    filter:{
      UniversityID:{
        _eq:"u1"}
    }){
    departments{
      head
    }
  }
}
```

(a) Query.

```
{
  "data":{
   "UniversityList":[
   {
    "departments":[
     {"head":"Harry,Potter"},
     {"head":"Sheldon,Cooper"}
    ]
   }]
  }
}
```

(b) Query Response.

Figure 3.3: An example GraphQL query/response.

3.3 Summary

In this chapter, we have introduced an overview of a GraphQL-based framework for data access and integration in which an ontology drives the generation of a GraphQL server. This framework can fill a gap in GraphQL applications in the respect of promoting GraphQL, not only for semantics-aware data access but for data integration, by automatically generating a GraphQL server based on ontologies and semantic mappings. The remaining chapters of this thesis are based on this framework and contribute to this framework in different perspectives. Next, we elaborate on the implementation of this framework, in terms of the GraphQL server generation, in Chapter 4.

Chapter

4

Ontology-based GraphQL server generation (OBG-gen)

We have introduced the outline of the GraphQL-based framework for data access and integration over multiple heterogeneous data sources in Chapter 3. In this chapter, we focus on how the GraphQL server generation is automated and move ahead to introduce our formal methods for generating GraphQL servers driven by ontologies. As part of the generation, in Section 4.1 we introduce a formal method for constructing a GraphQL schema based on an ontology. Then, in Section 4.2 we introduce our generic implementation of GraphQL resolver functions based on semantic mappings. In Section 4.3, we introduce the related work. Finally, we end the chapter with a summary in Section 4.4.

4.1 Ontology-based GraphQL schema generation

As mentioned in Section 3.1.1 of Chapter 3, the GraphQL schema represents a form of vocabulary supported by the GraphQL API rather than specifying what the data instances of an underlying data source may look like and what constraints have to be guaranteed. Therefore, we focus on GraphQL language features supporting semantics-aware and integrated data access, namely how data can be queried, rather than reflecting the semantics of a complex knowledge representation language in the context of a GraphQL schema. In Section 4.1.1, we introduce how a GraphQL schema is formalized. In Section 4.1.2, we introduce how an ontology is represented via a description logic TBox. Given an ontology represented in a description logic TBox, the concept and role names can be used to generate types and fields in a GraphQL schema, respectively. The relationships, which are represented as general concept inclusions in a description logic TBox can be used to specify how to connect generated types and fields in a GraphQL schema. Then, in Section 4.1.3, we present the core algorithm (*Schema Generator*) for generating a GraphQL schemas based on an ontology. In Section 4.1.4, we present the intended meaning of GraphQL schemas generated by the *Schema Generator*.

4.1.1 GraphQL schema formalization

According to [63, 64], a GraphQL schema can be defined over five finite sets. These five sets are $\mathtt{F} \subset \mathtt{Fields}$, $\mathtt{A} \subset \mathtt{Arguments}$, $\mathtt{T} \subset \mathtt{Types}$, $\mathtt{S} \subset \mathtt{Scalars}$, and $\mathtt{D} \subset \mathtt{Directives}$ where $\mathtt{T}$ is the disjoint union of $\mathtt{O}_\mathtt{T}$ (object types), $\mathtt{I}_\mathtt{T}$ (interface types), $\mathtt{U}_\mathtt{T}$ (union types), $\mathtt{IO}_\mathtt{T}$ (input object types) and $\mathtt{S}$. `Fields`, `Arguments`, `Types`, and `Directives` are pairwise disjoint, countably infinite sets representing field names, argument names, type names, and directive names, respectively. `Scalars`, which is a subset of `Types`, represents five built-in scalar types, which are `Int`, `Float`, `String`, `Boolean`, and `ID`. Moreover, the GraphQL schema definition language introduces *non-null types* and *list types*, called *wrapping types*, according to types in `Types`. Given a type `t` belonging to `Types`, the former is denoted as `t!`, while the latter is denoted as `[t]`. $\mathtt{W}_\mathtt{T}$ is used to denote the set of all types that can formed by wrapping the types in $\mathtt{T}$, and $\mathtt{W}_\mathtt{S}$ denotes the set of all types that can formed by wrapping the scalar types in $\mathtt{S}$. In our current work, considering the knowledge representation language we use for the ontology (see next section), we do not need directive and union types. Therefore, a GraphQL schema $\mathcal{S}$ is defined over $(\mathtt{F}, \mathtt{A}, \mathtt{T}, \mathtt{S})$ consisting of two assignments that are $type_\mathcal{S}$ and $implementation_\mathcal{S}$:

- $type_\mathcal{S} = type^\mathtt{F}_\mathcal{S} \cup type^\mathtt{AF}_\mathcal{S}$ where,
  - $type^\mathtt{F}_\mathcal{S} : (\mathtt{O}_\mathtt{T} \cup \mathtt{I}_\mathtt{T} \cup \mathtt{IO}_\mathtt{T}) \times \mathtt{F} \rightharpoonup \mathtt{T} \cup \mathtt{W}_\mathtt{T}$, which is a partial function since a type has a set of fields which is a subset of $\mathtt{F}$, assigns a type to each field that is defined for an object type, an interface type or an input object type,
  - $type^\mathtt{AF}_\mathcal{S} : dom(type^\mathtt{F}_\mathcal{S}) \times \mathtt{A} \rightharpoonup \mathtt{S} \cup \mathtt{W}_\mathtt{S} \cup \mathtt{IO}_\mathtt{T}$, which is a partial function since a field has a set of arguments which is a subset of $\mathtt{A}$, assigns a type to every argument of fields that are defined for a type;
- $implementation_\mathcal{S} : \mathtt{I}_\mathtt{T} \rightarrow 2^{\mathtt{O}_\mathtt{T} \cup \mathtt{I}_\mathtt{T}}$ assigns a set of object types or interface types to every interface type.

- $\mathtt{F} = \{\mathtt{UniversityID}, \mathtt{departments}, \mathtt{DepartmentID}, \mathtt{head}, \mathtt{AuthorID}, \mathtt{doctoralDegreeFrom}, \mathtt{\_and}, \mathtt{\_or}, \mathtt{\_not}, \mathtt{\_eq}, \mathtt{\_in}, \mathtt{\_neq}, \mathtt{\_nin}, \mathtt{\_like}, \mathtt{UniversityList}, \mathtt{DepartmentList}, \mathtt{AuthorList}, \mathtt{ProfessorList}\}$;
  $\mathtt{A} = \{\mathtt{filter}\}$;
  $\mathtt{T} = \mathtt{I}_\mathtt{T} \cup \mathtt{O}_\mathtt{T} \cup \mathtt{S} \cup \mathtt{U}_\mathtt{T} \cup \mathtt{IO}_\mathtt{T}$ where,
  - $\mathtt{I}_\mathtt{T} = \{\mathtt{Author}\}$,
  - $\mathtt{O}_\mathtt{T} = \{\mathtt{Query}, \mathtt{University}, \mathtt{Department}, \mathtt{Professor}\}$,
  - $\mathtt{S} = \{\mathtt{String}\}$,
  - $\mathtt{IO}_\mathtt{T} = \{\mathtt{UniversityFilter}, \mathtt{DepartmentFilter}, \mathtt{StringFilter}\}$;
- $type^{\mathtt{F}}_{\mathcal{S}} = \{(\mathtt{University}, \mathtt{UniversityID}) \mapsto \mathtt{String},$
  $(\mathtt{University}, \mathtt{departments}) \mapsto [\mathtt{Department}],$
  $(\mathtt{Department}, \mathtt{DepartmentID}) \mapsto \mathtt{String},$
  $(\mathtt{Department}, \mathtt{head}) \mapsto \mathtt{String},$
  $(\mathtt{Author}, \mathtt{AuthorID}) \mapsto \mathtt{String},$
  $(\mathtt{Professor}, \mathtt{AuthorID}) \mapsto \mathtt{String},$
  $(\mathtt{Professor}, \mathtt{doctoralDegreeFrom}) \mapsto [\mathtt{University}],$
  $(\mathtt{UniversityFilter}, \mathtt{UniversityID}) \mapsto \mathtt{StringFilter},$
  $(\mathtt{UniversityFilter}, \mathtt{departments}) \mapsto \mathtt{DepartmentFilter},$
  $(\mathtt{UniversityFilter}, \mathtt{\_and}) \mapsto [\mathtt{UniversityFilter}],$
  $(\mathtt{UniversityFilter}, \mathtt{\_or}) \mapsto [\mathtt{UniversityFilter}],$
  $(\mathtt{UniversityFilter}, \mathtt{\_not}) \mapsto \mathtt{UniversityFilter},$
  $(\mathtt{DepartmentFilter}, \mathtt{DepartmentID}) \mapsto \mathtt{StringFilter},$
  $(\mathtt{DepartmentFilter}, \mathtt{head}) \mapsto \mathtt{StringFilter},$
  $(\mathtt{DepartmentFilter}, \mathtt{\_and}) \mapsto [\mathtt{DepartmentFilter}],$
  $(\mathtt{DepartmentFilter}, \mathtt{\_or}) \mapsto [\mathtt{DepartmentFilter}],$
  $(\mathtt{DepartmentFilter}, \mathtt{\_not}) \mapsto \mathtt{DepartmentFilter},$
  $(\mathtt{StringFilter}, \mathtt{\_eq}) \mapsto \mathtt{String},$
  $(\mathtt{StringFilter}, \mathtt{\_in}) \mapsto [\mathtt{String}],$
  $(\mathtt{StringFilter}, \mathtt{\_neq}) \mapsto \mathtt{String},$
  $(\mathtt{StringFilter}, \mathtt{\_nin}) \mapsto [\mathtt{String}],$
  $(\mathtt{StringFilter}, \mathtt{\_like}) \mapsto \mathtt{String},$
  $(\mathtt{Query}, \mathtt{UniversityList}) \mapsto [\mathtt{University}],$
  $(\mathtt{Query}, \mathtt{DepartmentList}) \mapsto [\mathtt{Department}],$
  $(\mathtt{Query}, \mathtt{AuthorList}) \mapsto [\mathtt{Author}],$
  $(\mathtt{Query}, \mathtt{ProfessorList}) \mapsto [\mathtt{Professor}]\}$;
- $type^{\mathtt{AF}}_{\mathcal{S}} = \{((\mathtt{Query}, \mathtt{UniversityList}), \mathtt{filter}) \mapsto \mathtt{UniversityFilter},$
  $((\mathtt{Query}, \mathtt{DepartmentList}), \mathtt{filter}) \mapsto \mathtt{DepartmentFilter}\}$;
- $implementation_{\mathcal{S}} = \{\mathtt{Author} \mapsto \{\mathtt{Professor}\}\}$.

Figure 4.1: The formalization of the GraphQL schema shown in Figure 3.1.

Figure 4.1 illustrates a formalized representation of the GraphQL schema shown in Figure 3.1. In the formalization, we have sets $\mathtt{F}$, $\mathtt{A}$, $\mathtt{I_T}$, $\mathtt{O_T}$, $\mathtt{S}$ and $\mathtt{IO_T}$, which contains all the field names, argument names, interface type names, object type names, scalar type names and input object type names, respectively. Additionally, the formalization contains field declarations in the set $type^{\mathtt{F}}_{\mathcal{S}}$; argument declarations in $type^{\mathtt{AF}}_{\mathcal{S}}$; object types implementing interface types declarations in $implementation_{\mathcal{S}}$. For instance, $(\mathtt{University}, \mathtt{UniversityID}) \mapsto \mathtt{String}$ declares that the `University` type has a field `UniversityID` of which the returned type is `String`; $((\mathtt{Query}, \mathtt{UniversityList}), \mathtt{filter}) \mapsto \mathtt{UniversityFilter}$ declares that the `UniversityList` field accepts an input argument which is defined as the type `UniversityFilter`; $\mathtt{Author} \mapsto \{\mathtt{Professor}\}$ declares that the `Professor` type is one of the types that implement the interface `Author`.

4.1.2 Ontology represented by description logic TBox

In our work we assume that the ontology is represented by a TBox in a description logic, which is an extension of $\mathcal{FL}_0$ by adding qualified number restrictions. $\mathcal{FL}_0$ allows atomic concepts, the universal concept, the bottom concept, intersection and value restriction. This description logic can represent the semantics that can be reflected in a GraphQL schema for data access and integration. Figure 4.2 illustrates an example TBox for the university domain. Let $\mathsf{N_C}$, $\mathsf{N_R}$, $\mathsf{N_A}$, and $\mathsf{N_D}$ be disjoint finite sets of concept names, role names, attribute names, and datatype names respectively. For instance, in the example shown in Figure 4.2, we have four concept names `University`, `Department`, `Author`, and `Professor`; two role names `departments` and `doctoralDegreeFrom`; a datatype name `xsd:string`; and four attribute names `UniversityID`, `DepartmentID`, `head` and `AuthorID`. A TBox over $\mathsf{N_C}$, $\mathsf{N_R}$, $\mathsf{N_A}$ and $\mathsf{N_D}$ is a finite set of general concept inclusions (GCI) where each GCI is a statement in the form of $C \sqsubseteq E$, where C and E are concepts. We use a normalized TBox that contains only GCIs in the normal forms given in equation 4.1 where $A, B \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, $a \in \mathsf{N_A}$, and $d \in \mathsf{N_D}$, for generating the GraphQL schema. For instance, in the example shown in Figure 4.2, we have eight GCIs representing the relationship among concepts or relationships between concepts and datatypes. Normalization rules to obtain such a TBox are presented in [65]. The work in [66] shows that such normalization rules can preserve a conservative extension of a TBox in $\mathcal{FL}_0$.

A conservative extension guarantees that subsumptions with respect to the original TBox coincide with those with respect to the normalized TBox.

$$\begin{array}{lll} NF_1 : A \sqsubseteq B & NF_2 : A \sqsubseteq \forall r.B & NF_3 : A \sqsubseteq = 1 r.B \\ NF_4 : A \sqsubseteq \forall a.d & NF_5 : A \sqsubseteq = 1 a.d & \end{array} \quad (4.1)$$

$\mathsf{N_C}$ = {`University`, `Department`, `Author`, `Professor`}
$\mathsf{N_R}$ = {`departments`, `doctoralDegreeFrom`}
$\mathsf{N_D}$ = {`xsd:string`}
$\mathsf{N_A}$ = {`UniversityID`, `DepartmentID`, `head`, `AuthorID`}
`University` ⊑ ∀ `departments.Department`
`University` ⊑ =1 `UniversityID.xsd:string`
`Department` ⊑ =1 `DepartmentID.xsd:string`
`Department` ⊑ =1 `head.xsd:string`
`Author` ⊑ =1 `AuthorID.xsd:string`
`Professor` ⊑ `Author`
`Professor` ⊑ =1 `AuthorID.xsd:string`
`Professor` ⊑ ∀ `doctoralDegreeFrom.University`

Figure 4.2: An example TBox.

4.1.3 The *Schema Generator* algorithm

The details for generating a GraphQL schema are shown in Algorithm 1. An example input of the algorithm is shown in Figure 4.2. The output for the example is the schema shown in Figure 3.1. First, the algorithm iterates over the concept names in $\mathsf{N_C}$ (line 1 to line 5). For each concept, such as `University` in the example shown in Figure 4.2, the concept name (`University`) is used as the name of an object type to be generated (line 2); the term concatenated with ‘Filter’ is used as the name of an input type (`UniversityFilter`) to be generated (line 3); the term concatenated with ‘List’ is used as the name of a field (`UniversityList`) of the `Query` type (line 4). Additionally, each such field of the `Query` type is assigned an argument named ‘filter’, with a type that is the corresponding input type (line 5, e.g., `filter: UniversityFilter` to `UniversityList`). Next, the algorithm iterates over GCIs in the TBox (line 6 to line 30). For a GCI in the form of NF_1 (line 7 to line 12), the name of the super-concept is used as the name of an interface type to be generated (line 8); a field for the `Query` type named by concatenating the interface type

Algorithm 1: *Schema Generator*

```
Input  : $\mathsf{N_C}$; normalized TBox $\mathcal{TB}$;
         $\Phi$, mapping a datatype in $\mathsf{N_D}$ to a scalar type
Output: a GraphQL schema $\mathcal{S}$
1  for $A \in \mathsf{N_C}$ do
2      $\mathtt{O_T} = \mathtt{O_T} \cup \{A\}$ // extend $\mathcal{S}$ with an empty object type, $A$
3      $\mathtt{IO_T} = \mathtt{IO_T} \cup \{A\mathtt{Filter}\}$ // extend $\mathcal{S}$ with an empty input type, $A$Filter
       /* add following field/argument declarations to the Query type:
          $A$List(filter: $A$Filter): [$A$] */
4      $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(\mathtt{Query}, A\mathtt{List}) \mapsto [A]\}$
5      $type^{\mathtt{AF}}_{\mathcal{S}} = type^{\mathtt{AF}}_{\mathcal{S}} \cup \{((\mathtt{Query}, A\mathtt{List}), \mathtt{filter}) \mapsto A\mathtt{Filter}\}$
6  for $t \in \mathcal{TB}$ do
7      if $t$ is of the form $A \sqsubseteq B$   (i.e., $NF_1$) then
8          $\mathtt{I_T} = \mathtt{I_T} \cup \{B\}$ // extend $\mathcal{S}$ with an empty interface type, $B$
9          $\mathtt{IO_T} = \mathtt{IO_T} \cup \{B\mathtt{Filter}\}$ // extend $\mathcal{S}$ with an input type, $B$Filter
           /* add following field/argument declarations to the Query type:
              $B$List(filter: $B$Filter): [$B$] */
10         $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(\mathtt{Query}, B\mathtt{List}) \mapsto [B]\}$
11         $type^{\mathtt{AF}}_{\mathcal{S}} = type^{\mathtt{AF}}_{\mathcal{S}} \cup \{((\mathtt{Query}, B\mathtt{List}), \mathtt{filter}) \mapsto B\mathtt{Filter}\}$
12         $implementation_{\mathcal{S}}(B) = implementation_{\mathcal{S}}(B) \cup A$ // declare that the
             object type $A$ implements $B$
13     if $t$ is of the form of $A \sqsubseteq \forall r.B$   (i.e., $NF_2$) then
14         if $A \sqsubseteq_{=} 1r.B \in \mathcal{TB}$ then
15             Do nothing, this case will be handed in line 19 to line 21
16         else
               /* add following field declarations to $A$ and $A$Filter */
17             $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(A, r) \mapsto [B]\}$ // $r$: [$B$]
18             $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(A\mathtt{Filter}, r) \mapsto B\mathtt{Filter}\}$ // $r$: $B$Filter
19     if $t$ is of the form of $A \sqsubseteq_{=} 1r.B$   (i.e., $NF_3$) then
           /* add following field declarations to $A$ and $A$Filter */
20         $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(A, r) \mapsto B\}$ // $r$: $B$
21         $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(A\mathtt{Filter}, r) \mapsto B\mathtt{Filter}\}$ // $r$: $B$Filter
22     if $t$ is of the form of $A \sqsubseteq \forall a.d$   (i.e, $NF_4$) then
23         if $A \sqsubseteq_{=} 1a.d \in \mathcal{TB}$ then
24             Do nothing, this case will be handed in line 28 to line 30
25         else
               /* add following field declarations to $A$ and $A$Filter */
26             $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(A, r) \mapsto [\Phi(d)]\}$ // $r$: [$\Phi(d)$]
27             $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(A\mathtt{Filter}, r) \mapsto \Phi(d)\mathtt{Filter}\}$ // $r$: $\Phi(d)$Filter
28     if $t$ is of the form of $A \sqsubseteq_{=} 1a.d$   (i.e., $NF_5$) then
           /* add following field declarations to $A$ and $A$Filter */
29         $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(A, r) \mapsto \Phi(d)\}$ // $r$: $\Phi(d)$
30         $type^{\mathtt{F}}_{\mathcal{S}} = type^{\mathtt{F}}_{\mathcal{S}} \cup \{(A\mathtt{Filter}, r) \mapsto \Phi(d)\mathtt{Filter}\}$ // $r$: $\Phi(d)$Filter
```

name and 'List' is generated (line 10); the previously generated object type corresponding to the sub-concept implements the generated interface type (line 12).

From line 13 to line 21, the algorithm deals with GCIs containing roles (such as `University` $\sqsubseteq$ $\forall$ `departments.Department`), which can be of the form NF_2 or NF_3. In both cases, a field definition (e.g., `departments`) of the object type (e.g., `University`) and a field definition (`departments`) of the input type (`UniversityFilter`) are generated. However, for NF_3, the returned type of the field is defined as the original object type corresponding to the concept appearing on the right side of the GCI (line 20). For NF_2, the returned type is defined as a wrapped type, which is a list type (line 17). For instance, the `departments` field declaration for the `University` type is `departments:[Department]`. The algorithm deals with GCIs containing attributes in a similar way (line 22 to line 30). For example, the `University` object type has a field declaration, which is `UniversityID:String`. We define a function Φ for mapping a datatype that exists in the TBox to a scalar type in GraphQL. Due to the fact that current GraphQL supports five basic scalar types which are `ID`, `Float`, `Int`, `Boolean`, and `String`, our current implementation of function Φ focuses on mapping datatypes `xsd:float`, `xsd:int`, `xsd:string` and `xsd:boolean` to scalar types `Float`, `Int`, `String` and `Boolean`, respectively. However, GraphQL allows users to define custom scalar types, and the values of such custom types should be JSON serializable. Therefore, our Φ function can be easily extended in the future for mapping any datatype besides `xsd:float`, `xsd:int`, `xsd:string`, and `xsd:boolean` from a TBox into a custom scalar type in GraphQL.

Therefore, by generating the GraphQL schema based on an ontology we can, for each object or interface type and each field declaration, find the corresponding concept and relationship in the ontology. Since such concepts and relationships are used to define semantic mappings, when a resolver function (implemented based on semantic mappings) retrieves data sources of a requested type and relevant fields it can therefore understand the semantic mappings, which provide information regarding how to access underlying data sources and structure the returned data according to the GraphQL schema.

4.1.4 The intended meaning of GraphQL schemas generated by the *Schema Generator*

In Section 4.1.3, we present the *Schema Generator* which takes a TBox representing an ontology as an input, to generate a GraphQL schema. Such a GraphQL schema can describe how to access underlying data sources in which the data can be annotated by the ontology. The underlying data thus can be viewed as an ABox based on the TBox. Therefore, evaluating a GraphQL query conforming to this GraphQL schema can be viewed as retrieving the

ABox. Formally, an ABox, $\mathcal{A}$ is defined as a finite set of assertions of the form $C(x)$, $R(x,y)$ or $A(x,z)$, where $C \in \mathsf{N_C}, R \in \mathsf{N_R}, A \in \mathsf{N_A}$, x and y are instance names, z are literals. Figure 4.3 shows an example of ABox based on the TBox in Figure 4.2.

`University`(university_1), `University`(university_2);
`Department`(d1), `Department`(d2), `Department`(d3), `Department`(d4);
`departments`(university_1, d1), `departments`(university_1, d2),
`departments`(university_2, d3), `departments`(university_2, d4);
`UniversityID`(university_1, “u1”), `UniversityID`(university_1, “u2”);
`head`(d1, “Harry, Potter”), `head`(d2, “Sheldon, Cooper”),
`head`(d3, “Paul, Atredies”), `head`(d4, “Jack, Lee”).

Figure 4.3: An example ABox.

Definition 1. Let $\mathcal{Q}$ be a GraphQL query over (`F`, `A`, `T`, `S`), let $\mathcal{S}$ be a GraphQL schema over (`F`, `A`, `T`, `S`) such that $\mathcal{Q}$ conforms to $\mathcal{S}$. $\mathcal{S}$ is generated by the *Schema Generator* based on the TBox $\mathcal{T}$ representing the ontology $\mathcal{O}$. Let $\mathcal{D}$ be the underlying data that can be instantiated in terms of $\mathcal{O}$. Therefore, evaluating $\mathcal{Q}$ over $\mathcal{D}$ can be viewed as retrieving an ABox $\mathcal{A}$ based on $\mathcal{T}$:

- If $\mathcal{Q}$ requests an object or an interface type t with a field f of which the returned type is a scalar type s or the wrapping type $[s]$, in which $t \in \mathtt{O_T} \sqcup \mathtt{I_T}$, $f \in \mathtt{F}$, $s \in \mathtt{S}$, and $(t, f) \mapsto s \in type^{\mathtt{F}}_{\mathcal{S}}$ or $(t, f) \mapsto [s] \in type^{\mathtt{F}}_{\mathcal{S}}$, we can find the corresponding assertions in the ABox $\mathcal{A}$ of forms: $t(x)$ and $f(x,y)$;

- If $\mathcal{Q}$ requests an object or an interface type t_1 with a field f of which the returned type is another object or interface type t_2 or the wrapping type $[t_2]$, in which $t_1, t_2 \in \mathtt{O_T} \sqcup \mathtt{I_T}$, $f \in \mathtt{F}$, and $(t_1, f) \mapsto t_2 \in type^{\mathtt{F}}_{\mathcal{S}}$ or $(t_1, f) \mapsto [t_2] \in type^{\mathtt{F}}_{\mathcal{S}}$, we can find the corresponding assertions in the ABox $\mathcal{A}$ of forms: $t_1(x)$, $t_2(y)$ and $f(x,y)$.

For instance, given the GraphQL query shown in Figure 3.3a and the ABox shown in Figure 4.3, the following assertions are supposed to be retrieved: `University`(university_1), `departments`(university_1, d1), `departments`(university_1, d2), `head`(d1, “Harry, Potter”), `head`(d2, “Sheldon, Cooper”).

The above definition presents the meaning of the GraphQL schema generated based on a TBox for evaluating GraphQL queries. The definition relies on the *Schema Generator* where for each concept, the algorithm creates a corresponding type with the same name of the concept, same for roles and attributes. This guarantees to find the corresponding assertions from the ABox. However, in practice, as we presented in Section 3.1.2, how a GraphQL query can retrieve over the underlying data sources relies on how the resolver function is implemented when we construct GraphQL servers. In the next section, we present how resolver functions can be implemented in a generic way based on semantic mappings.

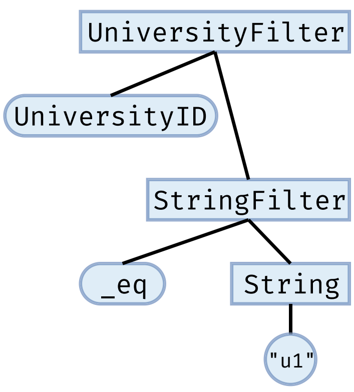
4.2 Generic GraphQL resolver function

In general, there are two styles for implementing resolver functions for a GraphQL server. One option is to implement one resolver function per type (object or interface) defined in the GraphQL schema, where such a function states how to fetch the data to populate relevant fields. For instance, since the `Query` type in Figure 3.1 has four field definitions (`UniversityList`, `DepartmentList`, `AuthorList`, and `ProfessorList`), we may provide four resolver functions for getting entities of the `University`, `Department`, `Author` and `Professor` types from underlying data sources, respectively. The other option is to provide a resolver function for every field of every type defined in the GraphQL schema, such that this resolver could return data for this field of any type. In our framework, we adopt the first style because it can be easily generalized based on semantic mappings. That is, we can implement just a generic resolver function that can be used to populate objects of any object type or interface type, and can be viewed as a built-in function of the GraphQL server. In Section 4.2.1, we introduce how a GraphQL query is represented by Abstract Syntax Trees (ASTs), in which one represents query fields and others represent the filter expression. Then in Section 4.2.2 we introduce the RDF Mapping Language (RML), which is used for representing semantic mappings. In Section 4.2.3, we describe the components of the generic resolver function. In Section 4.2.4, we present the core algorithm for the generic resolver function, which is responsible for accessing underlying data sources based on semantic mappings.

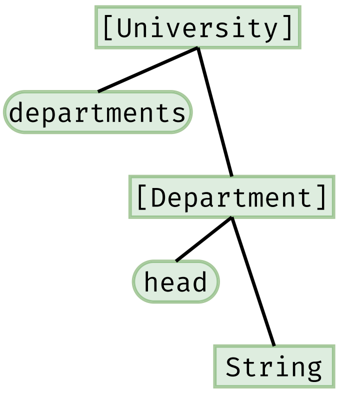
4.2.1 GraphQL queries represented by Abstract Syntax Trees

In general, a GraphQL query can be represented using a single Abstract Syntax Tree that contains nodes representing the fields requested in the query, and also contains additional nodes for the input arguments that may be used for each of these fields. In our approach, we assume that each query accepts an input argument which captures the notion of a filter condition. Therefore we specify the query evaluation in two steps: (i) evaluating for a filter condition, which is represented via an input argument that is defined as an input object type in the schema, (ii) evaluating for those fields that are requested in the GraphQL query. For instance, in the query example shown in Figure 3.3a, the field having a filtering condition is different from the requested fields (the former is `UniversityID` while the latter includes `departments` and `head`). In the evaluation step for the filter condition, the identifier information of the filtered out instances of the requested type (i.e., `University`) will be obtained after accessing the underlying data sources. In the next step, the underlying data sources will be accessed again to retrieve only the requested fields for the filtered instances. Therefore, to enable such two steps in the query evaluation, we use two ASTs to represent a GraphQL query (cf. Figure 4.4, these two ASTs represent the query shown in Figure 3.3a of Chapter 3), one of which captures the input argument structure (Figure 4.4a), and the other of which captures the structure of the query, including the requested fields and their types (Figure 4.4b). More specifically, every node in such ASTs represents either a type (i.e., object type, interface type, input type, or scalar type), a wrapping type, or a field. Additionally, ASTs that represent input arguments also contain nodes that represent the values of scalar-typed fields (e.g., `"u1"` in the AST shown in Figure 4.4a). The types (i.e., `UniversityFilter`, `StringFilter`, `String`) or wrapping types (i.e., `[University]`, `[Department]`) are drawn with rectangle nodes. The fields (i.e., `UniversityID`, `_eq`, `departments`, `head`) are drawn with rounded rectangle nodes.

In practice, a filter condition is converted into disjunctive normal form (DNF). DNF contains a sequence of disjuncts that are connected by the *OR* ($\lor$) operator, where each disjunct is a conjunction containing one or more terms connected by the *AND* ($\land$) operator [67, p. 633]. A query result satisfying DNF contains data formed by the union of data that satisfies each



(a) Abstract Syntax Tree for filter fields.



(b) Abstract Syntax Tree for query fields.

Figure 4.4: Abstract Syntax Trees for the query shown in Figure 3.3a.

disjunct (conjunction) [67, p. 633]. Therefore, in the step of evaluating for a filter condition: (i) multiple ASTs will be generated where each represents one of the conjunctions (disjuncts); (ii) the underlying data source will be accessed several times to filter out instances for each conjunction; (iii) a union of identifier information for the filtered out instances of the requested type will be returned.

4.2.2 RDF Mapping Language (RML)

RML [52, 53] is a declarative mapping language for linking data to ontologies [68]. An RML document has one or more `Triples Maps`, which declare how input data is mapped into triples of the form (subject, predicate, object). An example of RML mappings is shown in Listing 4.1. A `Triples Map` contains the following three components (`Logical Source`, `Subject Map` and a set of `Predicate-Object Maps`). A logical source declares the source of input data to be mapped. It contains definitions of `source` that locate the input data source, `reference formulation` declaring how to refer to the input data, and `logical iterator` declaring the iteration loop used to map the input data. For instance, line 2 to line 6 in Listing 4.1 constitute the definition of a logical source. The definition declares that the data source is a JSON-formatted data source on the Web and also describes the way of iterating the JSON-formatted data (line 5). A subject map declares a rule for generating subjects when transforming underlying data into triples, including how to construct URIs of subjects (e.g., line 8) and specifying the concept to which subjects belong (e.g., line 9). A predicate-object map consists of one or more predicate maps declaring how to generate predicates of triples (e.g., line 12), and one or more object maps or referencing object maps defining how to generate objects of triples. An object map can be a reference-valued term map or a constant-valued term map. The former declares a valid reference to a column (relational data sources), or to an object (JSON data sources). The latter declares the value of the object as constant data. For instance, line 39 to line 41 make up a reference-valued term map. Line 19 to line 25 constitute a definition of a referencing object map including the join condition based on two triples maps. A referencing object map refers to another triples map (called a parent triples map) by using a `rr:joinCondition` property to state the join condition between the current triples map and the parent triples map. A join condition contains two properties, `rr:child` and `rr:parent`, of which the values must be logical references to logical sources of the current triples map and the parent triples map, respectively.

Listing 4.1: An example of RML mappings transforming university domain data.

```
 1 <UniversityMapping>
 2  rr:logicalSource [
 3    rml:source "http://example.com/universities.json";
 4    rml:referenceFormulation ql:JSONPath;
 5    rml:iterator "$.data.universities[*]";
 6  ];
 7  rr:subjectMap [
 8    rr:template "http://example.com/university/{uid}";
 9    rr:class schema:University;
10  ];
11  rr:predicateObjectMap [
12    rr:predicate schema:UniversityID;
13    rr:objectMap [
14      rml:reference "uid";
15    ];
16  ];
17  rr:predicateObjectMap [
18    rr:predicate schema:departments;
19    rr:objectMap [
20      rr:parentTriplesMap <DepartmentMapping>
21      rr:joinCondition [
22       rr:child "uid";
23       rr:parent "university_id";
24      ];
25    ];
```

```
  ].

<DepartmentMapping>
  rr:logicalSource [
    rml:source "http://example.com/departments.csv";
    rml:referenceFormulation ql:CSV;
  ];
  rr:subjectMap [
    rr:template "http://example.com/department/{department_id}";
    rr:class schema:Department;
  ];
  rr:predicateObjectMap [
    rr:predicate schema:DepartmentID;
    rr:objectMap [
      rml:reference "department_id";
    ];
  ];
  rr:predicateObjectMap [
    rr:predicate schema:head;
    rr:objectMap [
     rml:reference "HEAD";
    ];
  ].
```

4.2.3 Components of the generic resolver function

We show the basic technical components of the generic resolver function including *QueryParser* and *Evaluator* in Figure 4.5.

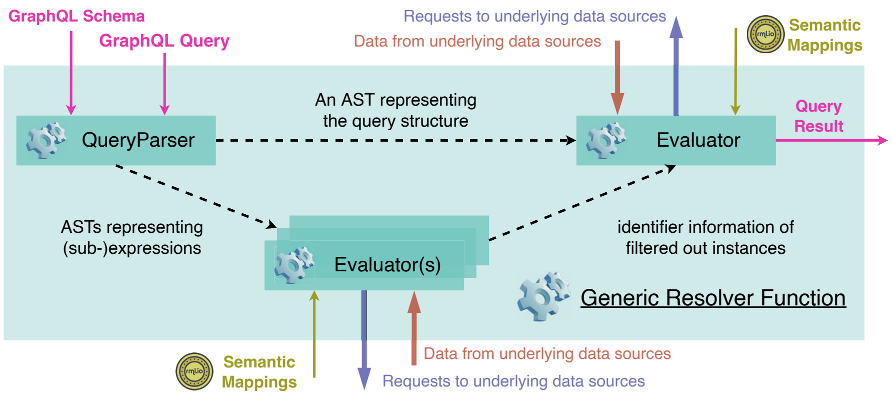

GraphQL Schema
GraphQL Query
Requests to underlying data sources
Data from underlying data sources
Semantic Mappings
An AST representing the query structure
QueryParser
Evaluator
Query Result
ASTs representing (sub-)expressions
Evaluator(s)
identifier information of filtered out instances
Generic Resolver Function
Semantic Mappings
Data from underlying data sources
Requests to underlying data sources

Figure 4.5: Technical components in the generic resolver function.

In Algorithm 2, we show the generic resolver function. The inputs to the generic resolver function are a GraphQL schema, a GraphQL query and semantic mappings. The GraphQL query and schema are inputs of the *QueryParser*. The *QueryParser* parses a query including a filter expression given as an input argument, and outputs the corresponding ASTs (e.g., Figure 4.4b) for the input argument and the query structure, respectively. As we mentioned in Section 4.2.1, in our practical solution a filter condition is converted into disjunctive normal form. As shown in Algorithm 2, the *QueryParser* parses the query, converts a filter expression into a union of conjunctive expressions, and generates an AST for each conjunctive expression and an AST for the query structure (line 2). Then, two processes, which are evaluating the filter expression (line 5 to line 7) and evaluating the query fields (line 9 and line 13), will continue. The *Evaluator* is responsible for sending requests to underlying data sources and fetching data according to an AST. During evaluation of the filter expression, for each AST representing a conjunctive

Algorithm 2: *Generic Resolver*

```
   Input : a GraphQL query: query; a GraphQL schema: schema;
           the semantic mappings: triples_maps
   Ouput: a list of objects of the type to be queried
1  Initialize an empty list: query_result
2  call QueryParser taking query and schema as inputs, to get ASTs for the
     filter condition and query fields: filter_asts, query_ast
3  if filter_asts is not Empty then
       /* there is an input argument given to the query */
4      Initialize an empty set: filtered_identifiers
5      for filter_ast in filter_asts do
6          call Evaluator taking filter_ast and triples_maps as inputs:
             identifier_info
7          merge filtered_identifiers and identifier_info:
             filtered_identifiers
8      if filtered_identifiers is not Empty then
9          call Evaluator taking query_ast, triples_maps and
             filtered_identifiers as inputs: query_result
10     else
11         Do nothing, there is not any instance from data sources satisfying
             the filter condition.
12 else
       /* there is not an input argument given to the query */
13     call Evaluator taking query_ast, triples_maps as inputs:
         query_result
14 return query_result
```

(sub-)expression, an evaluator is called to request data that satisfies the conjunctive (sub-)expression (line 6). After a call to an evaluator based on an AST (*filter_ast* in line 6), data representing the requested type, which contains identifier information, will be returned (*identifier_info* in line 6). Taking the query in Figure 3.3a represented by the ASTs shown in Figure 4.4 as an example, the requested type is `University` and data that can identify university instances is supposed to be returned in *identifier_info*. Such identifier information is captured in semantic mappings, which are used to construct the URIs for subjects where such subjects represent instances of the *University* concept. For instance, in line 8 of the RML mappings example in Listing 4.1, the values of the *uid* attribute of the underlying data source are used to construct URIs of subjects representing instances of the *University* concept. The identifier information returned by evaluating each *filter_ast* is merged into *filtered_identifiers* (line 7). During evaluation of the query fields, such merged identifier information is taken into account in the call to the evaluator of the query fields (line 9).

As we mentioned in Section 4.1.3, by generating the GraphQL schema based on an ontology, we can therefore, for each object or interface type and each field declaration, find the corresponding concept and relationship in the ontology. Since such concepts and relationships are used to define semantic mappings, when a generic resolver function retrieves data sources of a requested type and relevant fields, it can therefore understand the semantic mappings regarding how to access underlying data sources and structure the returned data according to the GraphQL schema. Taking the query in Figure 3.3a represented by the ASTs shown in Figure 4.4 as an example, as the requested type is `University`, the generic resolver function can therefore make use of relevant triples maps (line 1 to line 26 in Listing 4.1) defined in semantic mappings which are used for transforming underlying data following the semantics related to the *University* concept in the ontology.

4.2.4 The *Evaluator* algorithm

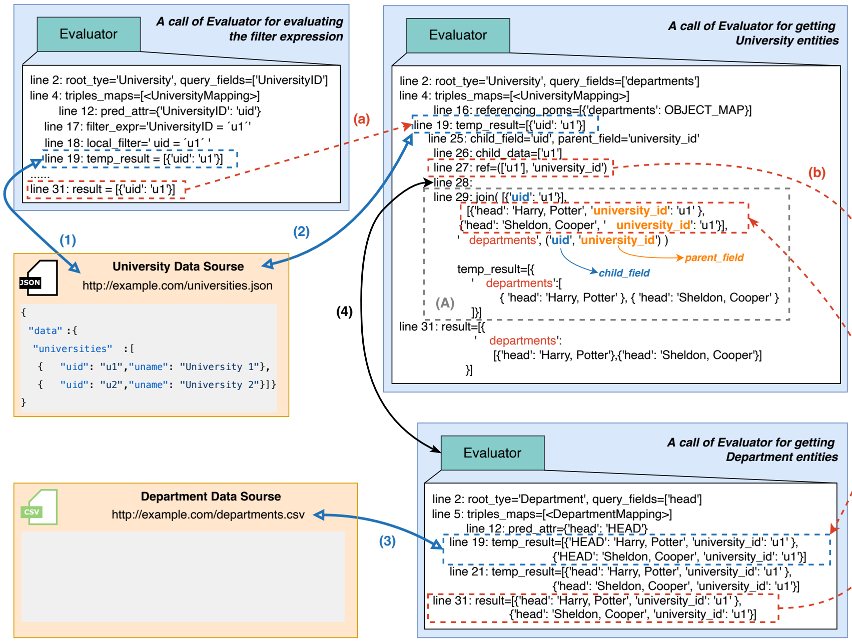
We present the details of *Evaluator* in Algorithm 3 and show an example in Figure 4.6 of how evaluators work for answering the query in Figure 3.3a. An AST and a number of triples maps from the semantic mappings are essential inputs to the algorithm. For a given AST, we can obtain the object type and fields that are requested in the query based on the root node and child nodes, respectively (line 2). For instance, taking the ASTs in Figure 4.4b as examples, the root type and the field for evaluating the filter expression are `University` and `UniversityID`, and the root type and the first level requested field for evaluating query fields are `University` and `departments`, respectively. After getting the relevant triples maps based on the root node type (line 4 in Algorithm 3, e.g., `UniversityMapping` in Listing 4.1) or from the argument (line 28, the parent triples map, `DepartmentMapping`, which is an argument in the recursive call of an evaluator), the algorithm iterates over triples maps and merges the data obtained based on each triples map (line 5 to line 30). Exploring this in more detail, the algorithm parses each triples map to get the logical source and relevant predicate-object maps (line 8 and line 9). As described in Section 4.2.2, there are three different types of predicate-object map depending on the different maps of object, which are a reference-valued term map, a constant-valued term map or a referencing-object map. The algorithm iterates over the predicate-object maps and parses each one (line 10 to line 16). For a reference-valued term map, the mapping between the predicate and the reference column or attribute is stored (line 12, e.g., `{UniversityID: uid}` is stored in *pred_attr*), which will be used for rewriting a filter expression according to the underlying data source (line 18, e.g., `uid = 'u1'`), annotating the obtained underlying data (line 21, e.g., `HEAD` is annotated as `head` for *Department* data). For a constant-valued term map, the mapping between the predicate and the constant data value and type is stored (line 14). Both *pred_attr* and *pred_const* will be used to annotate the data from underlying sources (line 21).

In the phase of evaluating a filter expression, *local_filter*, which represents the rewritten filter expression, is a necessary argument when sending requests to underlying data sources (line 19). While in the phase of evaluating query fields, *filter_ids*, being a *NULL* value or having at least one element, is a necessary argument (line 19, arrow `(a)` in Figure 4.6). A *NULL* value represents the fact that the GraphQL query does not include an input argument. After obtaining the data from the underlying data sources, the data is serialized into JSON format (key/value pairs) in which the keys are predicates stated in the predicate-object map (line 21), where each predicate corresponds to a field in the GraphQL schema. In the next step, the algorithm iterates over predicate-object maps in which the object map refers to another triples map (called a parent triples map) (line 22 to line 29). An evaluator is called again to fetch data based on this parent triples map (line 28, arrow `(4)` in Figure 4.6). For the query example, the parent triples map refers to the `DepartmentMapping`. Since such a referencing-object map definition states the join condition between the current triples map (`UniversityMapping`) based

Algorithm 3: *Evaluator*

```
Input  : an Abstract Syntax Tree: ast;
         the semantic mappings: triples_maps;
         the referencing data: ref;
         the identifiers for filtered out result: filtered_ids
Output: result of evaluating a filter expression or query fields
1  Initialize an empty list: result
2  get the root type and query fields from ast: root_type, query_fields
3  if triples_maps is Empty then
4      get relevant triples maps based on the root_type: triples_maps
5  for tm in triples_maps do
6      Initialize an empty list: referencing_poms
7      Initialize two empty lists: pred_attr, pred_const
8      get the logical source from tm: source
9      get all the predicate-object maps from tm based on query_fields: poms
10     for pom in poms do
11         if object_map in pom is a reference-valued term map then
12             extend pred_attr with a map between the predicate and
                 column/attribute
13         if object_map in pom is a constant-valued term map then
14             extend pred_const with a map between the predicate and data
                 value, type
15         if object_map is a referencing-object map term map then
16             extend referencing_poms with pom
17     parse ast and get the filter expression: filter_expr
18     localize filter_expr based on pred_attr: local_filter
19     access the data source based on source, local_filter, ref,
         filtered_ids: temp_result
20     if temp_result is not Empty then
21         annotate temp_result based on pred_attr, pred_const
22         for (pred, object_map) in referencing_poms do
23             get the sub tree from ast based on pred: sub_ast
24             parse object_map: parent_triples_map, join_condition
25             parse join_condition: child_field, parent_field
26             get the referencing data from temp_result on child_field:
                 child_data
27             ref = (child_data, parent_field)
28             call Evaluator based on sub_ast, parent_triples_map, ref:
                 parent_data
29             join temp_result and parent_data based on join_condition,
                 pred: temp_result
30     merge result and temp_result: result
31 return result
```

on *child_field* (`uid`) and the parent triples map (`DepartmentMapping`) based on *parent_field* (`university_id`) (line 21 to line 23 of the mappings in Listing 4.1), we can pass referencing data (*ref*), which contains the data obtained according to the current triples map and *parent_field*, to the call of an evaluator when we fetch data according to the parent triples map (line 28). Such referencing data is taken into account, in the recursive call to an evaluator, when the request is sent to the underlying data sources (line 19, arrow **(b)** in Figure 4.6). After the data is obtained according to the parent triples map (arrow **(c)** in Figure 4.6), it is joined with data obtained according to the current triples map (line 29, frame **(A)** in Figure 4.6).


Evaluator
A call of Evaluator for evaluating the filter expression
(a)
Evaluator
A call of Evaluator for getting University entities
(b)
(1)
(2)
(4)
(A)
child_field
parent_field
University Data Sourse
http://example.com/universities.json
JSON
Evaluator
A call of Evaluator for getting Department entities
Department Data Sourse
http://example.com/departments.csv
CSV
(3)
(c)

```
line 2: root_tye='University', query_fields=['UniversityID']
line 4: triples_maps=[<UniversityMapping>]
      line 12: pred_attr={'UniversityID': 'uid'}
   line 17: filter_expr='UniversityID = ´u1´'
   line 18: local_filter=' uid = ´u1´ '
   line 19: temp_result = [{'uid': 'u1'}]
......
line 31: result = [{'uid': 'u1'}]
```

```
line 2: root_tye='University', query_fields=['departments']
line 4: triples_maps=[<UniversityMapping>]
      line 16: referencing_poms=[{'departments': OBJECT_MAP}]
line 19: temp_result=[{'uid': 'u1'}]
   line 25: child_field='uid', parent_field='university_id'
    line 26: child_data=['u1']
    line 27: ref=(['u1'], 'university_id')
    line 28:
    line 29: join( [{'uid': 'u1'}],
           [{'head': 'Harry, Potter', 'university_id': 'u1' },
          {'head': 'Sheldon, Cooper', '  university_id': 'u1'}],
          '  departments', ('uid', 'university_id') )

          temp_result=[{
              '  departments':[
                    { 'head': 'Harry, Potter' }, { 'head': 'Sheldon, Cooper' }
                ]}]
line 31: result=[{
              '  departments':
                  [{'head': 'Harry, Potter'},{'head': 'Sheldon, Cooper'}]
            }]
```

```
{
 "data" :{
  "universities"  :[
   {   "uid": "u1","uname": "University 1"},
   {   "uid": "u2","uname": "University 2"}]}
}
```

```
line 2: root_tye='Department', query_fields=['head']
line 5: triples_maps=[<DepartmentMapping>]
      line 12: pred_attr={'head': 'HEAD'}
  line 19: temp_result=[{'HEAD': 'Harry, Potter', 'university_id': 'u1' },
                        {'HEAD': 'Sheldon, Cooper', 'university_id': 'u1'}]
  line 21: temp_result=[{'head': 'Harry, Potter', 'university_id': 'u1' },
                        {'head': 'Sheldon, Cooper', 'university_id': 'u1'}]
line 31: result=[{'head': 'Harry, Potter', 'university_id': 'u1' },
                 {'head': 'Sheldon, Cooper', 'university_id': 'u1'}]
```

Figure 4.6: An example for answering the query in Figure 3.3a, **(1)**-**(3)** indicate the requests to and responses from the data sources; **(a)**-**(c)** indicate the parameter passing between the calls to *Evaluators*; **(4)** indicates a recursive call to *Evaluator* for getting the data of *Departments*; frame **(A)** indicates a join operation.

4.3 Related work

The widely used Semantic Web-based techniques and the recently developed GraphQL have led to a number of works relevant to our GraphQL-based framework for data access and data integration. We extend the summary of approaches presented in [69] by adding several new related approaches and new perspectives on the comparison. Table 4.1 summarizes these systems and our approach. The majority of these systems can be divided into two categories, namely OBDA-based systems and GraphQL-based systems. The former group contains morph-rdb, morph-csv and Ontop. The latter group consists of GraphQL-LD, HyperGraphQL, UltraGraphQL, morph-graphql, Ontology2GraphQL and our OBG-gen. In addition to the two groups described above, there is also another system, OBA, which is an ontology-based framework that facilitates the development of REST APIs for knowledge graphs.

As a new perspective to the summary in [69], all the approaches (except for GraphQL-LD) have two processes: (i) the service setup (preparation) process and (ii) the query answering process. During the service setup process, some approaches need semantic mappings as input such as morph-rdb, morph-csv, Ontop, morph-graphql and OBG-gen. In such systems, semantic mappings are used in a similar manner to represent differences between global and local schemas. Morph-csv needs additional annotations for tabular data. OBG-gen needs an ontology and semantic mappings together in order to generate a GraphQL server that is intended not only for semantics-aware data access but for data integration. Morph-graphql requires semantic mappings to generate a GraphQL server intended for data access. Ontology2GraphQL needs a meta model for the GraphQL query language and requires an ontology following the meta model for generating the GraphQL schema. HyperGraphQL requires no inputs during the service setup process, but the developer must build the GraphQL server from scratch. UltraGraphQL, based on HyperGraphQL, requires RDF schemas of SPARQL endpoints for bootstrapping the GraphQL server. In actuality, GraphQL-LD does not require any GraphQL servers, but instead focuses on how to represent GraphQL queries using SPARQL algebra and to convert the results of a SPARQL query into a tree structure in response to a GraphQL query.

For the query answering process, OBDA-based approaches (i.e., morph-rdb, morph-csv and Ontop) accept SPARQL queries and translate them into SQL queries. Ontop and morph-rdb handle underlying data stored in relational databases, while morph-csv deals with data stored in CSV files. Our approach, OBG-gen, accepts relational data, CSV-formatted data and JSON-formatted data as the underlying data. The remaining approaches are based on underlying data in SPARQL endpoints and translate input queries (GraphQL queries for GraphQL-based approaches, API requests for OBA) into SPARQL queries. GraphQL-LD, HyperGraphQL, and UltraGraphQL require context information expressed in JSON-LD. Such JSON-LD context information contains URIs of classes to which instances in the RDF data belong.

4.4 Summary

In this chapter, we have elaborated on the implementation of the framework introduced in Chapter 3. We have also presented a formal method for generating a GraphQL schema based on an ontology. We then showed how a generic resolver function is implemented based on semantic mappings. In Section 4.3, we provided a detailed introduction to related work. Before we evaluate this framework and apply this framework in specific domains, we turn our focus to the preparation that is necessary to enable the usage of this framework. In other words, we focus on how to construct a domain ontology that will enable the GraphQL server generation process in the framework. Therefore, we introduce the development of a domain ontology for the materials design field (Chapter 5) and an approach for extending domain ontologies (Chapter 6 and Chapter 7). We present the evaluation of the framework in Chapter 8 and an application to the materials design field in Chapter 9.

Table 4.1: A summary of related approaches.

Approach	Service Setup (Preparation) Process		Query Answering Process		
	Input	Output	Input	Output	Underlying Data
morph-rdb [25, 70]	semantic mappings (R2RML)	–	SPARQL query	SQL query	Relational data
morph-csv [26]	semantic mappings (RML), tabular metadata	–	SPARQL query	SQL query	Tabular data
Ontop [27]	semantic mappings	–	SPARQL query	SQL query	Relational data
GraphQL-LD [71]	–	–	GraphQL query, JSON-LD context	SPARQL query	SPARQL endpoint
HyperGraphQL [72]	–	GraphQL server (manually)	GraphQL query, JSON-LD context	SPARQL query	SPARQL endpoint
UltraGraphQL [73, 74]	RDF schemas of SPARQL endpoints	GraphQL server (automatically)	GraphQL query, JSON-LD context	SPARQL query	SPARQL endpoint
morph-graphql [69]	semantic mappings (R2RML)	GraphQL server (automatically)	GraphQL query	SQL Query	Relational data
OBA [75]	an ontology	Open API specification; a REST API server (automatically)	API requests	SPARQL query	SPARQL endpoint
Ontology2GraphQL [76]	a meta model for GraphQL query language, an ontology follows the meta model	GraphQL server (automatically)	GraphQL query	SPARQL query	SPARQL endpoint
OBG-gen	semantic mappings (RML), an ontology	GraphQL server (automatically)	GraphQL query	**SQL query, API requests**	**Relational data, CSV-formatted data, JSON-formatted data**

Chapter

5

Materials Design Ontology (MDO)

For the framework presented in Chapter 3, a domain ontology plays an important role in generating a GraphQL server. Therefore, we turn our attention to domain ontology development for the materials design field, aiming not only to represent the domain knowledge but also to enable ontology-driven data access and integration. At the beginning of this work, no ontologies existed for the domain that could achieve such aims. In Section 5.1 we start by introducing an overview of background knowledge relevant to ontology development, and related work in the materials design field. In Section 5.2 we present the development of MDO (Materials Design Ontology), including the requirements analysis, and methodologies that were used. Then in Section 5.3, we introduce the concepts, relations, and the axiomatization of MDO. We also introduce the envisioned usage of MDO in Section 5.4, and summarize the impact, reusability, and availability of MDO in Section 5.5. Finally, the chapter concludes in Section 5.6.

5.1 Background and related work

Developing a domain-specific ontology for representing domain knowledge requires the developers to follow good practices of ontology development methodologies and to make good use of existing resources in relevant domains. In this Section, we first introduce several ontology engineering methodologies, one of which we use for developing MDO, then introduce an overview of existing ontologies, databases and an ongoing effort, OPTIMADE (*Open Databases Integration for Materials Design*), in the field.

5.1.1 Ontology development

Ontologies can support formalization to represent knowledge. However, the creation and management of ontologies do not come for free [77]. Therefore, the field *"Ontology Engineering"* studies the principles, methods and tools used for developing and maintaining ontologies [77]. Developing and maintaining an ontology is similar to software design in which software has a life cycle. Therefore, it is necessary to think about how to make the deliverables compatible and resilient in the life cycle. A variety of methods for ontology engineering have been developed by the community. The process of ontology development usually involves making a number of design choices. For instance, the background of the developers (ontology engineers or domain experts or both); the background knowledge taken into account (existing lexicons, thesauri, database schemas, or text such as interview transcripts); the tools for ontology development (engineering tools such as Protégé, evaluation and debugging tools such as OOPS! [78], RepOSE [79], management and versioning tools such as GitHub and Ontoology [80]). Many methodologies have been proposed for ontology development.

METHONTOLOGY is an early effort to develop a methodology for ontology engineering [81]. This methodology proposes that the life cycle of an ontology moves through states including *Specification*, *Knowledge Acquisition*, *Conceptualisation*, *Integration*, *Implementation*, *Evaluation*, and *Documentation*.

NeOn is a methodology for ontology engineering, proposing nine scenarios, which are commonly occurring situations, including *Scenario 1: From Specification to Implementation*, *Scenario 2: Reusing and re-engineering non-ontological resources*, *Scenario 3: Reusing ontological resources*, *Scenario 4: Reusing and re-engineering ontological resources*, *Scenario 5: Reusing and merging ontological resources*, *Scenario 6: Reusing, merging, and re-engineering ontological resources*, *Scenario 7: Reusing ontology design patterns (ODPs)*, *Scenario 8: Restructuring ontological resources*, and *Scenario 9: Localizing ontological resources* [82]. Depending on different existing background resources and the purpose of the ontology, developers can make use of different scenarios or combinations of scenarios from NeOn [82]. However, Scenario 1 should be included in any combinations since this scenario is a core activity that is necessary in the development of any ontology [82]. ***LOT*** [83, 84], (Linked Open Terms) is proposed based on NeOn methodology with a focus on matching the processes of ontology development with those of agile software development. LOT also focuses on reusing terms published in existing ontologies and reusing ontologies developed according to Linked Data principles.

On-To-Knowledge Methodology (OTKM) [85], focuses on constructing ontologies for knowledge management applications in enterprises, where such applications concern human issues, software engineering and the knowledge meta process. The knowledge meta process is similar to the definition and specification of ontology development activities. Within this knowledge meta process, there are several activities including *Feasibility Study*, *Kickoff*, *Refinement*, *Evaluation*, and *Application & Evolution*.

Along with the methodologies provided above, Ontology Design Patterns (ODPs) provide another method for guiding the development of ontologies. A representative ODP-based ontology development methodology is ***eXtreme Design*** [86]. This methodology focuses on incremental development, inspired by the eXtreme Programming (XP) agile software development approach. The idea of eXtreme Design is that ODPs representing generic use cases can be matched against local use cases defined in the requirements of the ontology to be developed. Thus, an important part of such a methodology is selecting existing ODPs that are suitable.[1] Moreover, the work in [87] presents how to integrate ontology matching and debugging processes into the incremental development process of eXtreme Design.

We chose NeOn to guide the development of MDO. In particular, we focused on applying scenario 1 (*From Specification to Implementation*), scenario 2 (*Reusing and re-engineering non-ontological resources*), scenario 3 (*Reusing ontological resources*) and scenario 8 (*Restructuring ontological resources*). We did not consider the other scenarios because of we design MDO for semantics-aware and integrated querying over materials databases and it is only necessary to reuse certain concepts from other ontological resources. We did not need to re-engineer or merge other ontological resources. Although we could have used approaches such as eXtreme Design [86] or its extension [87] which are modern approaches in terms of considering ontology design patterns, on-

[1]A repository of ODPs is available at `http://ontologydesignpatterns.org`.

tology matching and debugging, since our initial ontology is expected to be of a smaller size and given our earlier experience with the NeOn methodology for ontology engineering, we decided to use NeOn. In addition, none of existing ontology design patterns were suitable for reuse in MDO to achieve semantics-aware and integrated querying. NeOn allows combinations of scenarios covering different activities that might be involved in the life cycle of an ontology, in contrast to rigid settings of workflows from other methodologies such as METHONTOLOGY, OTKM [82]. In addition, it considers (i) the collaborative aspects of ontology development and (ii) the reuse and dynamic evolution of ontology networks [82]. In the materials science domain, we see the trend that different domain ontologies are emerging. It is foreseeable that the materials science field will need and have a large number of domain ontologies assembling ontology networks that use different resources for development and that are developed collaboratively by different people. Therefore, following NeOn methodology to develop MDO permits us to consider all the necessary aspects of ontology development which would be needed in the future maintenance and extension of MDO.

5.1.2 Ontologies in the materials science domain

A number of ontologies in the materials science field have been developed and we show some characteristics in Table 5.1 from knowledge representation and materials science perpectives. EMMO (earlier known as European Materials & Modelling Ontology, and recently renamed Elementary Multiperspective Material Ontology)[2] is a top-level ontology with the purpose of developing a standard representational ontology framework based on knowledge of materials modeling and characterization. Most other ontologies, however, are domain ontologies that focus on specific sub-domains of the materials science field (*Domain* column in Table 5.1) and have been developed with a specific use in mind (*Application Scenario* column in Table 5.1). MatOnto [88], based on the top-level ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering),[3] aims to represent structured knowledge, properties and processing steps relevant to materials for data exchange, reuse and integration. MatOWL [89] is extracted from MatML schema data to enable ontology-based data access. The latter, MatML,[4] is an extensible markup

[2]https://github.com/emmo-repo/EMMO

[3]http://www.loa.istc.cnr.it/dolce/overview.html

[4]https://www.matml.org

language (XML) for exchanging materials information. The Materials Ontology in [90] is designed for data exchange among thermal property databases, particularly focusing on representing knowledge relevant to material processing, measurement methods and manufacturing processes. The NanoParticle Ontology [91], based on the Basic Formal Ontology (BFO)[5] [92], and the eNanoMapper ontology [93] are two ontologies in the nanotechnology domain. The former represents properties of nanoparticles to design new nanoparticles, while the latter focuses on assessing risks caused by the use of nanomaterials in engineering. Extensions to these ontologies are computed in [4] and are presented in Chapter 7. The MMOY ontology [94] captures metal materials knowledge from Yago. The Materials and Molecules Basic Ontology (MAMBO) [95] reuses some concepts and relationships in MDO and focuses on materials based on molecules. The Dislocation Ontology [96] focuses on representing knowledge related to crystalline materials and reuses some concepts from MDO. The Platform MaterialDigital Ontology (PMD) [97] is a prototype to describe materials science experiments.

The Materials Design Ontology (the last row in Table 5.1 of which we introduce more details in the rest of the chapter), aims to enable semantic and integrated querying over multiple heterogeneous materials databases, which cannot be fulfilled by the other ontologies. They are either designed for specific domains (e.g., MatOnto for crystals) or are designed as a top-level ontology (i.e., EMMO) which contains semantics that are not necessary for semantic and integrated querying over multiple materials databases.

From the knowledge representation perspective, the basic terms defined in these ontologies shown in Table 5.1 involve materials, properties, performance, and processing in specific sub-domains. All of the ontologies presented use OWL as a representation language (*Language* column in Table 5.1). The number of OWL classes ranges from a few to several thousands (*Ontology Metrics* column in Table 5.1). Some ontologies have more classes than properties (e.g., MatOnto, Materials Ontology, NanoParticle Ontology, MMOY and EMMO), while some have many more properties (e.g., MDO). Several ontologies are developed in a modular fashion (*Modularity* column in Table 5.1).

[5]`http://basic-formal-ontology.org/`

Table 5.1: Characteristics of main ontologies in the materials science field.

Ontology	Knowledge Representation Perspective			Materials Science Perspective	
	Ontology Metrics	Language	Modularity	Domain	Application Scenario
EMMO	309 classes, 35 properties, 3 individuals	OWL	✓	Materials science	Top-level ontology
MatOnto [88]	78 classes, 10 properties, 24 individuals	OWL	✓	Crystals	Materials discovery
MatOWL [89]	(not available)	OWL		Materials	Semantic querying
Materials Ontology [90]	606 classes, 31 properties, 488 individuals	OWL	✓	Thermal properties	Data exchange, search
ELSSI-EMD ontology [98]	35 classes, 37 properties, 33 individuals	OWL	✓	Materials testing	Standardization
NanoParticle Ontology [91]	1904 classes, 81 properties	OWL		Nanotechnology	Data integration, search
eNanoMapper [93]	12781 classes, 5 properties 464 individuals	OWL	✓	Nanotechnology	Data integration
MMOY [94]	2325 classes, 9 properties, 1738 individuals	OWL		Metals	Knowledge extraction
MAMBO [95]	26 classes, 33 properties	OWL	✓	Molecules-based materials	Knowledge representation
Dislocation Ontology [96]	18 classes, 16 properties	OWL	✓	Crystalline materials	Knowledge representation
PMD [97]	13 classes, 7 properties	OWL	✓	Materials experiments	Knowledge representation, Data curation
MDO [3]	37 classes, 64 properties	OWL	✓	Materials design	Semantic/Integrated querying over multiple databases

5.1.3 Databases in the materials science domain

The Inorganic Crystal Structure Database (ICSD) [99] is a frequently utilized database for completely identified inorganic crystal structures, with nearly 200k entries [100, 101]. The data contained in ICSD serves as an important starting point in many electronic structure calculations. Several other crystallographic information resources are also available [102]. A popular open access resource is the Crystallography Open Database (COD) [103] with nearly 400k entries [104]. Closely related to COD is the Predicted Crystallography Open Database (PCOD) [105] with over 1 million predicted crystal structures. Another open access resource that relates to COD is the Theoretical Crystallography Open Database (TCOD) [106] with 2,906 entries. A number of databases for phase identification are hosted at the International Centre for Diffraction Data (ICDD) [107]. These databases have been in use by experimentalists for a long time. Springer Materials Springer Materials [108] contains, among many other data sources, the well-known Landolt Börnstein database, an extensive data collection from many areas of physical sciences and engineering. The Japan National Institute of Materials Science (NIMS) Materials Database MatNavi [109] contains a wide collection of mostly experimental but also some computational electronic structure data. Thermodynamical data, which is necessary for computing phase diagrams with the CALPHAD method, exists in many different databases [110]. Open access databases with relevant data can be found through OpenCalphad [111].

Databases of results from electron structure calculations have existed in some form for several decades. In 1978, Moruzzi, Janak, and Williams published a book with computed electronic properties such as density of states, bulk modulus and cohesive energy of all metals [112]. It is only in the last few years, however, that the idea of collecting computed results at a large scale in publicly available databases for general has become widespread. Prominent examples of databases or repositories that appeared early during the present trend are the Electronic Structure Project (ESP) [113], the Automatic Flow for Materials Discovery (AFLOW) [114, 115], the Materials Project [116, 16], the Open Quantum Materials Database (OQMD) [17, 18], and the Novel Materials Discovery (NOMAD) [20]. There is now a growing demand for open science from funding agencies, regulatory bodies, the scientific community and the general public. Data management plans are becoming mandatory, and making research data, also raw data, available is now expected and becoming the norm in research. This has lead to an explosion of available materials science datasets and archived data of varying quality and usefulness. Many of the above mentioned repositories have made their frameworks available (e.g., Automated Interactive Infrastructure and Database for Computational Science (AiiDA) [117, 118], the Atomic Simulation Environment (ASE) [119, 120], and the high-throughput toolkit (httk) [121, 122]).

5.1.4 Open Databases Integration for Materials Design

OPTIMADE [123] is a consortium that gathers many database providers and has made a first stable release of an API specification in 2021. The majority of the databases are listed in Section 5.1.3. It aims at enabling interoperability among materials databases through a common REST API. During the development of OPTIMADE, widely used materials databases such as those introduced in Section 5.1.3 were taken into account. OPTIMADE maintains a schema that defines the specification of the OPTIMADE API. The OPTIMADE API specification includes, essentially, a list of terms for which there is a consensus from different database providers. For the development of MDO, these terms serve as a basis. Such terms mainly concerns structural information about materials, with limited representation of semantic relationships among these terms.

5.2 Development of Materials Design Ontology

We use OWL2 DL as the representation language for MDO. During the entire process, two knowledge engineers, and one domain expert from the materials design domain were involved. In the remainder of this section, we introduce the key aspects of the development of MDO.

5.2.1 Requirements analysis

Since we developed MDO from scratch, the requirements analysis is the first step and a core activity in *Scenario 1: From Specification to Implementation.* In the context of ontology development, requirements analysis is usually represented as competency questions, restrictions, reasoning requirements classified as functional requirements, and non-functional requirements such as naming conventions, documentation and extendibility. During this step, we clarified the requirements by proposing use cases (UC), competency questions (CQ) and additional restrictions (AR).

5.2.1.1 Use cases

The use cases, which were identified through literature study and discussion between the domain expert and the knowledge engineers based on experience with the development of OPTIMADE and the use of materials science databases, are listed below.

- UC1: MDO will be used for representing knowledge in basic materials science such as solid-state physics and condensed matter theory.
- UC2: MDO will be used for representing materials calculation and standardizing the publication of the materials calculation data.
- UC3: MDO will be used as a standard to improve the interoperability among heterogeneous databases in the materials design domain.
- UC4: MDO will be mapped to the schema of OPTIMADE to improve the search functionality of OPTIMADE.

5.2.1.2 Competency questions

The competency questions are based on discussions with domain experts and contain questions that the materials databases (as listed in Section 5.1.3) generally do not provide an easy way to answer as well as questions that experts would want to ask the databases. For instance, CQ1, CQ2, CQ6, CQ7, CQ8 and CQ9 cannot be asked explicitly via the database APIs, although the original downloadable data contains the answers. The SPARQL queries that correspond to the following competency questions are given in Appendix A.

- CQ1: What are the calculated properties and their values produced by a materials calculation?
- CQ2: What are the input and output structures of a materials calculation?
- CQ3: What is the space group type of a structure?
- CQ4: What is the lattice type of a structure?
- CQ5: What is the chemical formula of a structure?
- CQ6: For a series of materials calculations, what are the compositions of materials with a specific range of a calculated property (e.g., band gap)?

- CQ7: For a specific material and a given range of a calculated property (e.g., band gap), what is the lattice type of the structure?
- CQ8: For a specific material and an expected lattice type of output structure, what are the values of calculated properties of the calculations?
- CQ9: What is the computational method used in a materials calculation?
- CQ10: What is the value for a specific parameter (e.g., cutoff energy) of the method used for the calculation?
- CQ11: Which software produced the result of a calculation?
- CQ12: Who are the authors of the calculation?
- CQ13: Which software or code does the calculation run with?
- CQ14: When was the calculation data published to the database?

5.2.1.3 Additional restrictions

Further, we proposed a list of additional restrictions that help in defining concepts. Some examples are shown below.

- AR1: A materials property can relate to a structure.
- AR2: A materials calculation has exactly one corresponding computational method.
- AR3: A structure corresponds to one specific space group.
- AR4: A materials calculation is performed by some software program or code.
- AR5: A structure is a part of some materials.
- AR6: A calculation is achieved by a specific computational method.
- AR7: A structure and a property can be published by references which could be databases or publications.
- AR8: A calculation can take some structures as input.
- AR9: A calculation can take some properties as input.

5.2.2 Using existing resources

Developing an ontology does not mean redefining everything. As the second scenario of the NeOn methodology presents, reusing and re-engineering non-ontological resources are activities that avoid *"reinventing the wheel"* [82].

Such non-ontological resources could be thesauri, glossaries or databases in the domain of interest. During the development of MDO, we have had discussions with the domain expert regarding the scope of concepts and relationships to be modeled in MDO after the requirements analysis, as well as the selection of relevant non-ontological resources. We then analyzed these selected non-ontological resources and decided how to make use of them in the development of MDO. These resources are: (i) the dictionaries of CIF (Crystallographic Information Framework)[6] and International Tables for Crystallography,[7] and (ii) the APIs from different databases (e.g., Materials Project, AFLOW, OQMD) and OPTIMADE. The former helps in modeling concepts and relationships relevant to materials structural knowledge that is involved in the requirements analysis (e.g., UC1, CQ3, AR3). The latter helps in modeling concepts and relationships relevant to materials calculation knowledge (e.g., UC2, CQ1, AR2).

In the next step, we took a look at the third scenario of NeOn methodology, which is reusing ontological resources, and make use of some existing ontological resources in the development of MDO. We started by searching, assessing and comparing existing ontologies (as described in Section 5.1.2 and shown in Table 5.1). We reused the concept *Material* from EMMO and the concept *atom* from ChEBI (Chemical Entities of Biological Interest) [124]. EMMO is a top-level ontology for the materials science field and *Material* is a general concept in it, which can connect to other domain ontologies. Reusing the *Material* concept in MDO makes it possible to connect MDO with other domain ontologies. ChEBI contains a conceptualization of knowledge of chemical elements, which is useful for modeling materials composition relevant knowledge in MDO (e.g., CQ5). As we present in the requirements analysis, MDO needs to represent numerical values for materials properties, and provenance information of materials calculations. Therefore, we reused the *Quantity*, *QuantityValue*, *QuantityKind* and *Unit* concepts, as well as relevant relationships from QUDT (Quantities, Units, Dimensions and Data Types Ontologies) [125], and the *Agent* and *SoftwareAgent* concepts and relevant relationships from PROV-O [126]. Additionally, for ontology annotation, we used the metadata terms from the Dublin Core Metadata Initiative (DCMI)[8] to represent the metadata of MDO.

[6]`https://www.iucr.org/resources/cif`
[7]`https://it.iucr.org`
[8]`http://purl.org/dc/terms/`

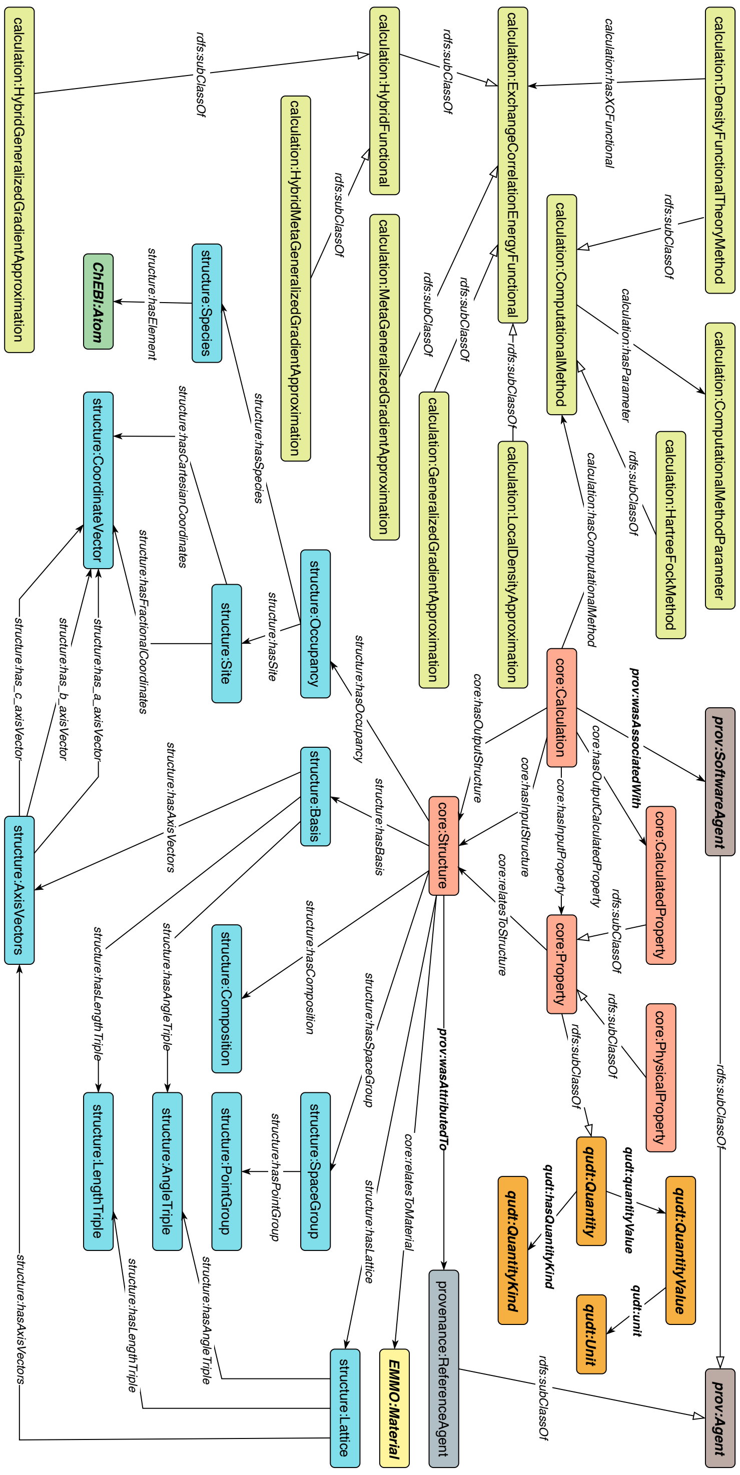


Figure 5.1: An overview of MDO.

5.3 Description of Materials Design Ontology

MDO consists of one basic module, *Core*, and two domain-specific modules, *Structure* and *Calculation*, which imports the core module. In addition, the *Provenance* module, which also imports *Core*, models provenance information. In total, the OWL2 DL representation of the ontology contains 37 classes, 32 object properties, and 32 data properties. Figure 5.1 shows an overview of the ontology. The ontology specification is also publicly accessible at w3id.org.[9] The competency questions can be answered using the concepts and relations in the different modules (CQ1 and CQ2 by *Core*, CQ3 to CQ8 by *Structure*, CQ9 and CQ10 by *Calculation*, and CQ11 to CQ14 by *Provenance*).

5.3.1 MDO core module

The **Core** module as shown in Figure 5.2, consists of the top-level concepts and relations of MDO, which are also reused in other modules. Figure 5.3 shows the description logic axioms for the core module. The module represents general information about materials calculations. The concepts *Calculation* and *Structure* represent materials calculations and materials structures, respectively, while *Property* represents materials' properties. *Property* is specialized into the disjoint concepts *CalculatedProperty* and *PhysicalProperty* (Core1, Core2, Core3). *Property*, which can be viewed as a quantifiable aspect of one material or materials system, is defined as a subconcept of *Quantity* from QUDT (Core4). *Properties* are also related to *structures* (Core5). When a calculation is applied to materials structures, each *calculation* takes some *structures* and *properties* as input, and may output *structures* and *calculated properties* (Core6, Core7). In addition, we reuse the concept *Material* of EMMO and state that each *structure* is related to some *material* (Core8).

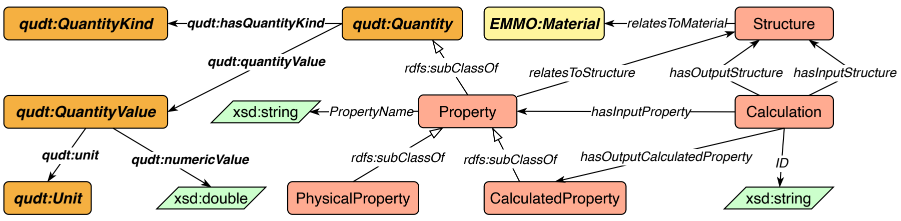


Figure 5.2: Concepts and relations in the Core module.

[9]https://w3id.org/mdo/full/1.0/

(Core1) CalculatedProperty $\sqsubseteq$ Property
(Core2) PhysicalProperty $\sqsubseteq$ Property
(Core3) CalculatedProperty $\sqcap$ PhysicalProperty $\sqsubseteq \bot$
(Core4) Property $\sqsubseteq$ Quantity
(Core5) Property $\sqsubseteq \forall$ relatesToStructure.Structure
(Core6) Calculation $\sqsubseteq \exists$ hasInputStructure.Structure
$\sqcap \forall$ hasInputStructure.Structure $\sqcap \forall$ hasOutputStructure.Structure
(Core7) Calculation $\sqsubseteq \exists$ hasInputProperty.Property $\sqcap \forall$ hasInputProperty.Property
$\sqcap \forall$ hasOutputCalculatedProperty.CalculatedProperty
(Core8) Structure $\sqsubseteq \exists$ relatesToMaterial.Material $\sqcap \forall$ relatesToMaterial.Material

Figure 5.3: Description logic axioms for the Core module.

5.3.2 MDO structure module

The **Structure** module as shown in Figure 5.4, represents the structural information of materials. Figure 5.5 shows the description logic axioms for the structure module. Each *structure* has exactly one *composition*, which represents the chemical elements that compose the structure and the ratio of elements in the *structure* (Struc1). The *composition* has different representations of chemical formulas. The *occupancy* of a structure relates the *sites* with the *species*, i.e., the specific chemical elements, which occupy the *site* (Struc2–Struc5). Each *site* has at most one representation of coordinates in Cartesian format and at most one in fractional format (Struc6, Struc7). The spatial information regarding structures is essential to reflect physical characteristics such as melting point and strength of materials. To represent this spatial information, we state that each *structure* is represented by some *bases* and a (periodic) *structure* can also be represented by one or more *lattices* (Struc8). Each *basis* and each *lattice* can be identified by one *axis-vectors* set or one *length triple* together with one *angle triple* (Struc9, Struc10). An *axis-vectors* set has three connections to *coordinate vector* representing the coordinates of three translation vectors respectively, which are used to represent a (minimal) repeating unit (Struc11). These three translation vectors are often called *a*, *b*, and *c*. Point groups and space groups are used to represent information of the symmetry of a structure. The *space group* is the group of symmetry operations that map the *structure* to itself. Of these operations, subgroups that keep at least one point fixed form the *point groups*. The *space group* represents a symmetry group of patterns in three dimensions of a *structure* and the *point group* represents a group of linear mappings, which correspond to the group of motions in space to determine the symmetry of a *structure*. Each *structure* has one corresponding *space group* (Struc12). Based on the definition from International Tables for Crystallography, each *space group* also has some corresponding *point groups* (Struc13).

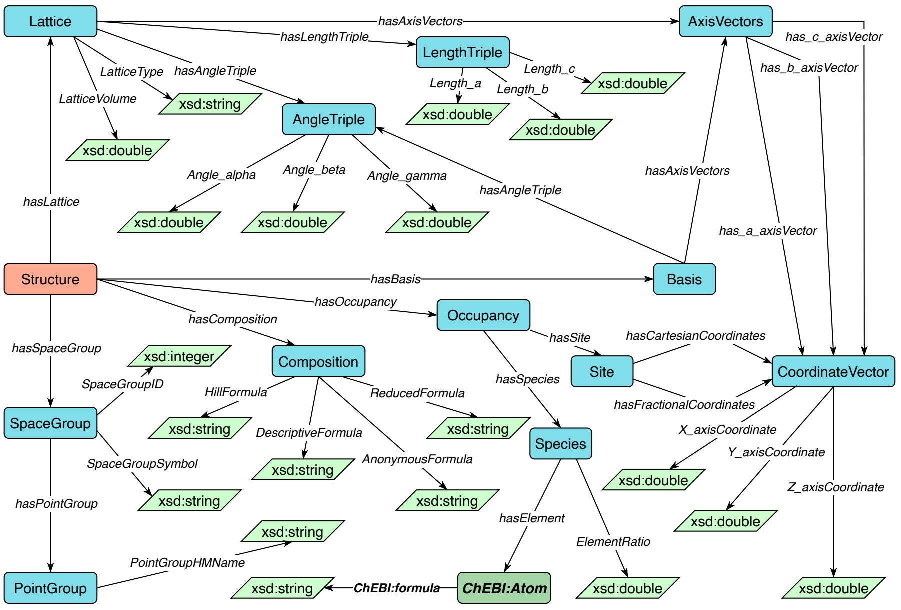


Figure 5.4: Concepts and relations in the Structure module.

5.3.3 MDO calculation module

The **Calculation** module as shown in Figure 5.6, represents the classification of different computational methods. Figure 5.7 shows the description logic axioms for the *Calculation* module. Each *calculation* is achieved via a specific *computational method* (Cal1). Each *computational method* has some *parameters* (Cal2). In the current version of this module, we represent two different methods, the *density functional theory method* and the *HartreeFock method* (Cal3, Cal4). In particular, the density functional theory method is frequently used in materials design to investigate the electronic structure. Such a method has at least one corresponding *exchange correlation energy functional* (Cal5), which is used to calculate the exchange-correlation energy of a system. There are different kinds of functionals to calculate exchange–correlation energy (Cal6–Cal11).

(Struc1) Structure $\sqsubseteq$ $= 1$ hasComposition.Composition
$\sqcap$ $\forall$ hasComposition.Composition
(Struc2) Structure $\sqsubseteq$ $\exists$ hasOccupancy.Occupancy $\sqcap$ $\forall$ hasOccupancy.Occupancy
(Struc3) Occupancy $\sqsubseteq$ $\exists$ hasSpecies.Species $\sqcap$ $\forall$ hasSpecies.Species
(Struc4) Occupancy $\sqsubseteq$ $\exists$ hasSite.Site $\sqcap$ $\forall$ hasSite.Site
(Struc5) Species $\sqsubseteq$ $= 1$ hasElement.Atom
(Struc6) Site $\sqsubseteq$ ≤ 1 hasCartesianCoordinates.CoordinateVector
$\sqcap$ $\forall$ hasCartesianCoordinates.CoordinateVector
(Struc7) Site $\sqsubseteq$ ≤ 1 hasFractionalCoordinates.CoordinateVector
$\sqcap$ $\forall$ hasFractionalCoordinates.CoordinateVector
(Struc8) Structure $\sqsubseteq$ $\exists$ hasBasis.Basis $\sqcap$ $\forall$ hasBasis.Basis $\sqcap$ $\forall$ hasLattice.Lattice
(Struc9) Basis $\sqsubseteq$ $= 1$ hasAxisVectors.AxisVectors $\sqcup$
($= 1$ hasLengthTriple.LengthTriple $\sqcap$ $= 1$ hasAngleTriple.AngleTriple)
(Struc10) Lattice $\sqsubseteq$ $= 1$ hasAxisVectors.AxisVectors $\sqcup$
($= 1$ hasLengthTriple.LengthTriple $\sqcap$ $= 1$ hasAngleTriple.AngleTriple)
(Struc11) AxisVectors $\sqsubseteq$ $= 1$ has_a_axisVector.CoordinateVector
$\sqcap$ $= 1$ has_b_axisVector.CoordinateVector
$\sqcap$ $= 1$ has_c_axisVector.CoordinateVector
(Struc12) Structure $\sqsubseteq$ $= 1$ hasSpaceGroup.SpaceGroup
$\sqcap$ $\forall$ hasSpaceGroup.SpaceGroup
(Struc13) SpaceGroup $\sqsubseteq$ $\exists$ hasPointGroup.PointGroup
$\sqcap$ $\forall$ hasPointGroup.PointGroup

Figure 5.5: Description logic axioms for the Structure module.

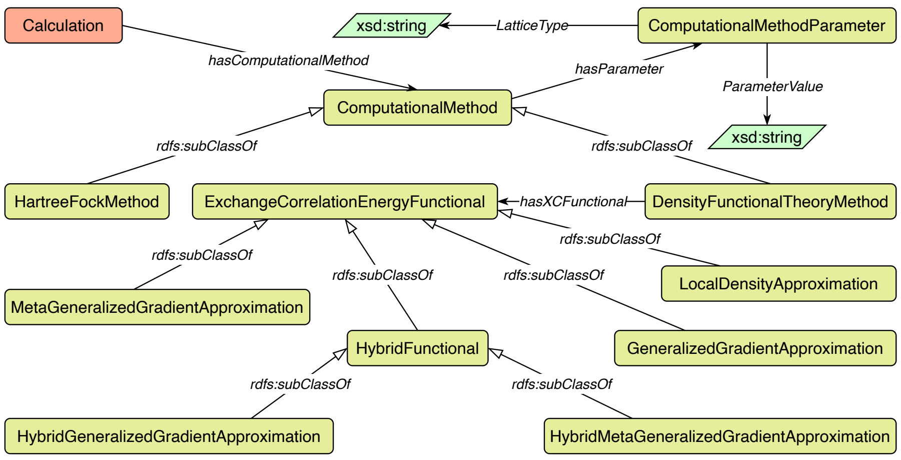


Figure 5.6: Concepts and relations in the Calculation module.

(Cal1) Calculation $\sqsubseteq\, = 1$ hasComputationalMethod.ComputationalMethod
(Cal2) ComputationalMethod $\sqsubseteq \exists$ hasParameter.ComputationalMethodParameter
$\sqcap\ \forall$ hasParameter.ComputationalMethodParameter
(Cal3) DensityFunctionalTheoryMethod $\sqsubseteq$ ComputationalMethod
(Cal4) HartreeFockMethod $\sqsubseteq$ ComputationalMethod
(Cal5) DensityFunctionalTheoryMethod $\sqsubseteq$
$\exists$ hasXCFunctional.ExchangeCorrelationEnergyFunctional
$\sqcap\ \forall$ hasXCFunctional.ExchangeCorrelationEnergyFunctional
(Cal6) GeneralizedGradientApproximation $\sqsubseteq$ ExchangeCorrelationEnergyFunctional
(Cal7) LocalDensityApproximation $\sqsubseteq$ ExchangeCorrelationEnergyFunctional
(Cal8) MetaGeneralizedGradientApproximation $\sqsubseteq$
ExchangeCorrelationEnergyFunctional
(Cal9) HybridFunctional $\sqsubseteq$ ExchangeCorrelationEnergyFunctional
(Cal10) HybridGeneralizedGradientApproximation $\sqsubseteq$ HybridFunctional
(Cal11) HybridMetaGeneralizedGradientApproximation $\sqsubseteq$ HybridFunctional

Figure 5.7: Description logic axioms for the Calculation module.

5.3.4 MDO provenance module

The **Provenance** module, as shown in Figure 5.8, represents the provenance information of materials data and calculations. Figure 5.9 shows the description logic axioms for the *Provenance* module. We reuse part of PROV-O and define a new concept *ReferenceAgent* as a sub-concept of the *Agent* concept PROV-O (Prov1). We state that each *structure* and *property* can be published by *reference agents*, which could be databases or publications (Prov2, Prov3). Each *calculation* is produced by a specific *software* (Prov4).

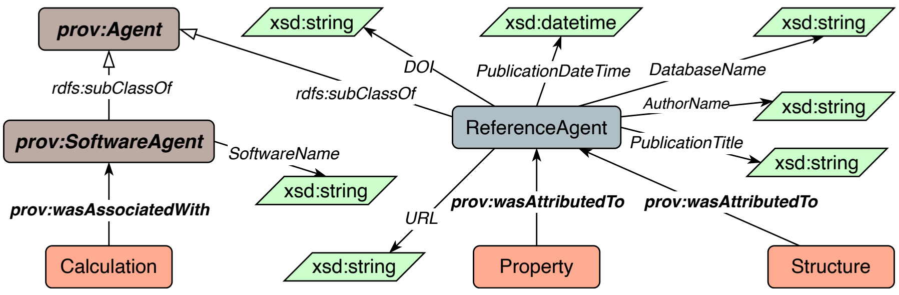


Figure 5.8: Concepts and relations in the Provenance module.

(Prov1) ReferenceAgent ⊑ Agent
(Prov2) Structure ⊑ ∀ wasAttributedTo.ReferenceAgent
(Prov3) Property ⊑ ∀ wasAttributedTo.ReferenceAgent
(Prov4) Calculation ⊑ ∃ wasAssociatedwith.SoftwareAgent

Figure 5.9: Description logic axioms for the Provenance module.

5.4 Usage of Materials Design Ontology

In Figure 5.10, we show the envisioned use of MDO for semantic search over OPTIMADE and materials science databases. As we introduced in Section 2.2 of Chapter 2, there are two ways to implement an ontology-based data access approach: one is to materialize the underlying data to RDF data so that the data can be queried by SPARQL queries; the other is to virtually access the underlying data based on the semantic mappings. Using MDO can enable both kinds of ontology-based data access approaches. By defining mappings between MDO and the schemas of materials databases, we can create MDO-enabled query interfaces. The querying can occur, for instance, via MDO-based query expansion, MDO-based mediation or through MDO-enabled data warehouses. In Figure 5.10, the process labeled with `(a)-(e)` shows the materialized way of accessing data. The process labeled with `(1)-(4)` shows a virtual way of accessing data, which is similar to the framework presented in Chapter 3. In addition, we provide an example to show how MDO can represent the domain knowledge by instantiating a materials calculation using MDO terminology, in Section 5.4.1.

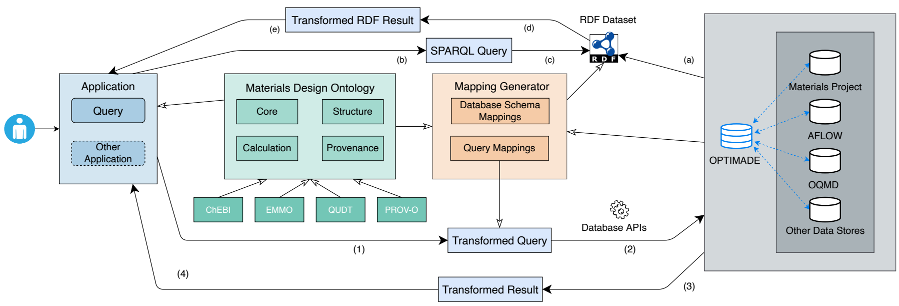


Figure 5.10: The envisioned use of MDO, `(a)-(e)` indicate ontology-based data access in a materalized way; `(1)-(4)` indicate a virtual way of data access by which the framework presented in Chapter 3 follows.

5.4.1 Instantiating a materials calculation using MDO

In Figure 5.11 we exemplify the use of MDO to represent a specific materials calculation and related data in an instantiation. The example is from one of the 85 stable materials published in the Materials Project in [121]. The calculation is about one kind of elpasolites, with the composition $Rb_2Li_1Ti_1Cl_6$. To avoid overcrowding the figure, we only show the instances corresponding to the output structure of the calculation, and for multiple calculated properties, species and sites, we only show one instance respectively. Connected to the instances of the core module's concepts are instances representing the structural information of the output structure, the provenance information of the output structure and calculated properties, and the information about the computational method used for the calculation.

5.5 Impact, reusability, and availability of MDO

To our knowledge, MDO is the first ontology representing concepts and relationships relevant to solid-state physics, which are the basis for materials design. The ontology fills a need for semantically enabling access to and integration of materials databases, and for realizing FAIR data in the materials design field. This will have a large impact on the effectiveness and efficiency of finding relevant materials data and calculations, thereby augmenting the speed and the quality of the materials design process. Through our connection with OPTIMADE and because of the fact that we have created mappings between MDO and some major materials databases, the potential for impact is significant.

The development of MDO followed well-known practices from the ontology engineering point of view (NeOn methodology and modular design). We also reused some concepts from PROV-O, ChEBI, QUDT and EMMO. A permanent URL[10] is reserved from w3id.org for MDO. MDO is maintained on a GitHub repository,[11] from which the ontology in OWL2 DL, visualizations of the ontology and modules, UCs, CQs and restrictions are available. It is licensed via an MIT license.[12] Due to our modular approach MDO can be extended with other modules, for instance, regarding different types of calculations and their specific properties. Several other efforts on building specific

[10]https://w3id.org/mdo/
[11]https://github.com/LiUSemWeb/Materials-Design-Ontology
[12]https://github.com/LiUSemWeb/Materials-Design-Ontology/blob/master/LICENSE

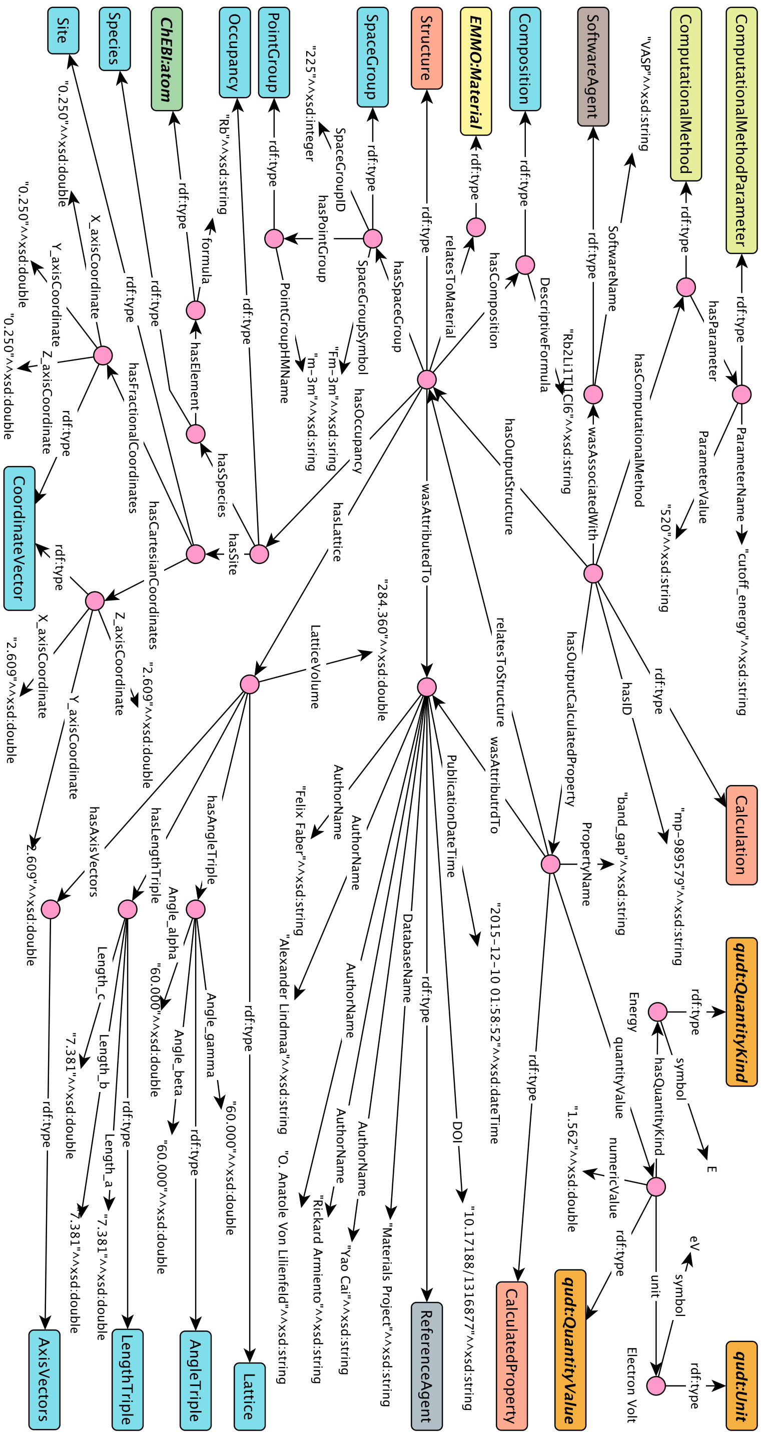


Figure 5.11: An instantiated materials calculation.

domain ontologies such as MAMBO [95] and Dislocation Ontology [96] reuse some concepts from MDO.

5.6 Summary

In this chapter, we have introduced the background in terms of ontologies, databases in materials design domain, and an effort (OPTIMADE) that is intended to integrate data in the field. We have introduced the details of the development of MDO, which is inspired by OPTIMADE. MDO is an essential output in the scope of the framework introduced in Chapter 3 in terms of employing the GraphQL-based framework for data access and integration in the materials design field. Moreover, in the next chapter, we focus on introducing a method for extending domain ontologies of which we make use to produce candidates for extending two ontologies in the nanotechnology domain, as well as MDO.

Chapter

6 An approach for extending domain ontologies (ToPMine-FTCA)

In the framework presented in Chapter 3, when new databases are added or new kinds of data are added to existing databases, the coverage of the ontology driving the GraphQL server generation may need to be enlarged. Therefore, we study how ontologies can be extended and propose an approach based on phrase-based topic modeling, formal topical concept analysis and domain expert validation. The phrase-based topic model (ToPMine) is used to mine unstructured text of interest. The formal topical concept analysis (FTCA) is used to derive the relationships among topics and obtain more specific formal topical concepts. Domain experts provide validation on the result of the phrase-based topic modeling in terms of frequent phrases and topics, and on the result of the formal topical concept analysis in terms of formal topical concepts. This chapter is organized as below. In Section 6.1 we introduce the relevant background knowledge. In Section 6.2, we introduce the framework of our approach (ToPMine-FTCA). Finally, we summarize the chapter in Section 6.3.

6.1 Background

We begin by introducing how ontologies can be extended by mining unstructured text in Section 6.1.1. Then, we introduce topic models in Section 6.1.2.

6.1.1 Extending ontologies based on unstructured text

The ontology extension problem that we tackle in this thesis deals mainly with concept discovery and concept hierarchy derivation. These are also two of the tasks in the problem of ontology learning [127]. Therefore, most of the related work comes from that area. For instance, a recent survey [128] discusses 140 research papers. Different techniques can be used for concept and relationship extraction. In this setting, new ontology elements are derived from text using knowledge acquisition techniques.

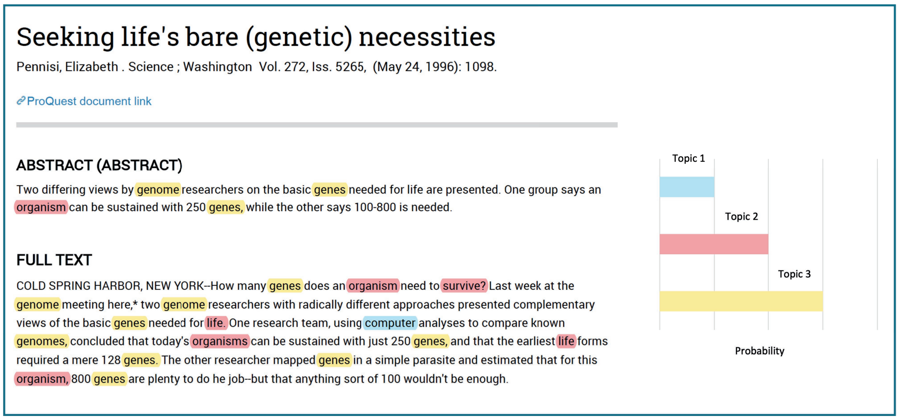
Linguistic techniques use part-of-speech tagged corpora for extracting syntactic structures that are analyzed regarding the words and the modifiers contained in the structure. One kind of linguistic approach is based on linguistics using lexico-syntactic patterns. The pioneering research conducted in this line is in [129], which defines a set of patterns indicating is-a relationships between words in the text. Other linguistic approaches may make use of, for instance, compounding, the use of background and itemization, term co-occurrence analysis or superstring prediction (e.g., [130, 131]).

Another paradigm is based on machine learning and statistical methods, which use the statistics of the underlying corpora, such as the k-nearest neighbors approach [132], association rules [133], bottom-up hierarchical clustering techniques [134], supervised classification [135] and formal concept analysis [136]. There are also some approaches that use topic models [137, 138, 139] but they focus on concept names that are words, rather than phrases. In [140], a phrase-based topic model is proposed in which each topic is represented by a number of phrases. Ontology evolution approaches [141, 142] allow for the study of changes in ontologies and using the change management mechanisms to detect candidate missing relations. An approach that allows for detection and user-guided completion of the is-a structure is given in [143, 144], where completion is formalized as an abduction problem and the RepOSE tool is presented.

We chose topic models as the basis for mining unstructured text in our work due to the fact that topic models have the ability to generate the abstract information from a collection of documents, which is valuable when deriving new concepts, axioms or relationships for extending ontologies. Moreover, we chose the phrase-based topic model in [140] for the reason that it can discover topical phrases of arbitrary length, which is more interesting for representing domain knowledge in materials design field. Based on our study of existing ontologies and databases for the materials science field as shown in Chapter 5, we observed that the conceptualization in these ontologies and the schemas of these databases contain terms expressed by more than one meaningful word. Therefore it is advantageous to use the phrase-based topic model in [140].

6.1.2 Topic models

A topic model is a statistical model that represents the abstract topics expressed in a collection of documents and the most common topic model takes Latent Dirichlet Allocation (LDA) [145] as the basis, which can be easily represented by its generative process. Given a collection of documents, words are generated by the generative process in two stages: (i) a distribution over topics is drawn randomly, (ii) for each word in the document, first choose a topic randomly from the result in stage (i), and then choose a word from the corresponding distribution over the vocabulary [146].


Seeking life's bare (genetic) necessities

Pennisi, Elizabeth . Science ; Washington Vol. 272, Iss. 5265, (May 24, 1996): 1098.

ProQuest document link

ABSTRACT (ABSTRACT)

Two differing views by genome researchers on the basic genes needed for life are presented. One group says an organism can be sustained with 250 genes, while the other says 100-800 is needed.

FULL TEXT

COLD SPRING HARBOR, NEW YORK--How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do he job--but that anything sort of 100 wouldn't be enough.

Figure 6.1: The intuitions behind Latent Dirichlet Allocation for representing a collection of documents [146].

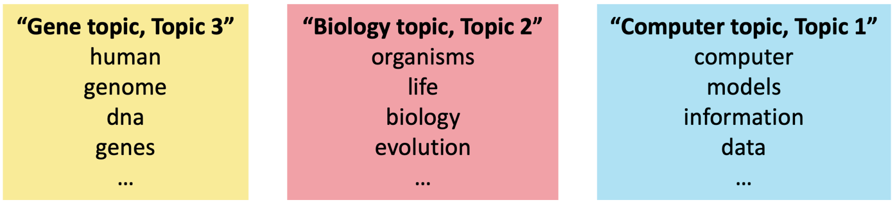


Figure 6.2: An example of the inference with Latent Dirichlet Allocation [146].

Figure 6.1 illustrates an example of how a topic model views a document. From the perspective of a topic model, this document on the subject of *"Seeking Life's Bare (Genetic) Necessities"* belongs to a number of topics such as the gene topic marked in yellow, the biology topic in red and the computer topic in blue. The document belongs to each topic to a certain degree, as shown in Figure 6.1. To represent the document, we can draw the categorical probability distribution over a number of topics. Meanwhile, each topic can be represented by a list of words which are more strongly correlated with the topic, as shown in Figure 6.2. For instance, the gene topic includes words such as *human*, *genome* and *dna* with high rankings in the list. A common topic model can be viewed as working based on unigrams, while the phrase-based topic models concern topical phrases of mixed lengths. ToPMine [140] is one of the methods that consider phrases. ToPMine has a combination of a frequent phrase mining framework, which segments a document, and a topic modeling method, which works on the document partition.

6.2 The framework (ToPMine-FTCA)

The framework for extending ontologies, shown in Figure 6.3, contains the following steps. In the first step, *creation of a phrase-based topic model*, documents related to the domain of interest are used to create topics. The phrases, as well as the topics, are suggestions that a domain expert should validate or interpret and relate to concepts in the ontology. In the second step the (possibly validated and updated) topics are used in a *formal topical concept analysis*, which returns suggestions to the domain expert regarding relations between topics and thus concepts in the ontology. Both steps lead to the addition of new concepts and (subsumption) axioms to the ontology. In the following subsections we describe these steps.

6.2.1 Topic model-based text mining

In our first step we use the phrase-based topic model, ToPMine [140]. Given a corpus of documents and the number of requested topics, representations of latent topics in the documents are generated by ToPMine. Essentially, topics can be seen as a probability distribution over words or phrases. ToPMine is purely data-driven, i.e., it does not require domain knowledge or specific linguistic rule sets. This is important for an application domain (e.g., the

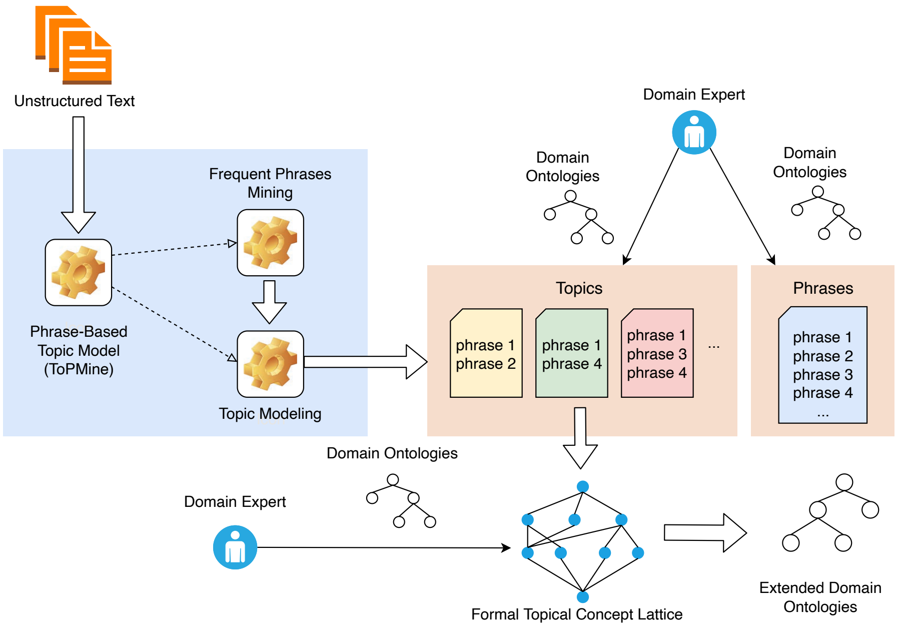


Figure 6.3: Approach: The upper part of the Figure shows the creation of a phrase-based topic model with as input unstructured text and as output phrases and topics. The lower part shows the formal topical concept analysis with as input topics and as output a topical concept lattice. In both parts a domain expert validates and interprets the results.

materials design domain) as there is a lack of annotated background knowledge. An important property of the system is that it works on bag-of-phrases, rather than the traditional bag-of-words. This means that words occurring closer together have more weight than words that are further away. Also, as we assume existing ontologies, it is very likely that concepts with one-word names are already in the ontology, and so we focus on phrases.

ToPMine consists of two parts: frequent phrases mining and topic modeling. In the first part, frequent contiguous phrases are mined, which consists of collecting aggregate counts for all contiguous words satisfying a minimum support threshold. Then the documents are segmented based on the frequent phrases, and an agglomerative phrase construction algorithm merges the frequent phrases guided by a significance score. In the second part, topics are generated using a variant of Latent Dirichlet Allocation, called PhraseLDA, which deals with phrases rather than words. For instance, if ToPMine is applied to mine the document shown in Figure 6.1, the generated topics can have phrases as representatives (e.g., *'computer analyses'* could be generated to represent the computer topic).

6.2.2 Formal topical concept analysis

After we obtain results from ToPMine, we define a new variant of formal concept analysis (e.g., [147]) and use this new variant on topics. These topics can come directly from the previous step or can be a modified version of the topics of the previous step, where non-relevant topics or phrases have been removed.

We first define the notions of formal topical context, formal topical concept and topical concept lattice.[1]

Definition 2 (Formal Topical Context)**.** A formal topical context is a triple (P, T, I) where P is a set of phrases, T is a set of topics, and I is a binary relation between P and T $(I \subseteq P \times T)$.

We can also refer to the elements of P as objects and those of T attributes. For instance, in the example shown in Figure 6.4, the set P consists of five phrases, while the set T consists of five topics. The binary relation I indicates phrase occurrences in topics.

Definition 3 (Formal Topical Concept)**.** (A, B) is a formal topical concept of (P, T, I) iff $A \subseteq P$, $B \subseteq T$, $A' = B$, $B' = A$ where $A' := \{t \in T \mid \forall p \in A : < p, t > \in I\}$ and $B' := \{p \in P \mid \forall t \in B : < p, t > \in I\}$. A is the extent and B is the intent of (A, B).

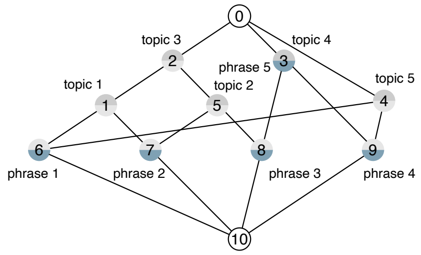
In this definition, A' is the set of all topics common to the phrases of A. In the other way, B' is the set of all phrases that have all topics in B. For instance, in the example shown in Figure 6.4, we have a formal topical concept *({phrase 1, phrase 2}, {topic 1, topic 3})*. That means the two topics are common to the two phrases, and vice versa.

Definition 4 (Topical Concept Lattice)**.** Topical formal concepts can be ordered. We say that $(A_1, B_1) \leq (A_2, B_2)$ iff $A_1 \subseteq A_2$. The set $\Phi(P, T, I)$ of all formal topical concepts of (P, T, I), with this order, is called the topical concept lattice of (P, T, I).

[1]Note that formal topical concepts should not be confused with concepts in the ontologies.

	topic 1	topic 2	topic 3	topic 4	topic 5
phrase 1	✓		✓		✓
phrase 2	✓	✓	✓		
phrase 3			✓	✓	
phrase 4				✓	✓
phrase 5				✓	

(a) Example of phrase occurrences in topics



(b) Example of Formal Topical Concept Lattice

ID	*Formal Topical Concept (FTC)*	*ID*	*Formal Topical Concept (FTC)*
0	({phrase 1, phrase 2, phrase 3, phrase 4, phrase 5}, {})	6	({phrase 1}, {topic 1, topic 3, topic 5})
1	**({phrase 1, phrase 2}, {topic 1, topic 3})**	7	({phrase 2}, {topic 1, topic 2, topic 3})
2	({phrase 1, phrase 2, phrase 3}, {topic 3})	8	({phrase 3}, {topic 2, topic 3, topic 4})
3	({phrase 3, phrase 4, phrase 5}, {topic 4})	9	({phrase 4}, {topic 4, topic 5})
4	({phrase 1, phrase 4}, {topic 5})	10	({}, {topic 1, topic 2, topic 3, topic 4, topic 5})
5	({phrase 2, phrase 3}, {topic 2, topic 3})		

(c) Formal Topical Concepts

Figure 6.4: Examples of (a) phrase occurrences in topics, (b) Formal Topical Concept Lattice and (c) Formal Topical Concepts.

In the lattice shown in Figure 6.4, a node represents a formal topical concept. For a formal topical concept (A, B), its extent (a set of phrases) is found by collecting all phrases in its node as well as its descendants. The intent (a set of topics) is found by collecting all topics in its node as well as its ancestors.

6.2.3 Domain expert validation

As shown in Figure 6.3, a domain expert is involved in the different steps in our approach to validate and interpret the results of the phrase-based topic model and the formal topical concept analysis.

The domain expert **validates or interprets all frequent phrases** that appear in all topics, which are outputs of ToPMine. The outcome can be one of the following.

- (i) A frequent phrase is a meaningful representation of a concept in the specific domain and it is already in the ontology. For example, *gold nanoparticle* is a specific concept within the nanotechnology domain and it is already in the NanoParticle Ontology. We distinguish two cases: (i) a concept with the same name or a name that is a synonym of the original form of the frequent phrase already exists in the ontology ('EXIST') or (ii) a concept with a name that is a modified form of the frequent phrase already exists in the ontology ('EXIST-m').

- (ii) A frequent phrase is a meaningful representation of a concept in the specific domain but it is not in the ontology. For example, *microcrystalline silicon* is a meaningful representation of a concept but such a concept does not exist in the ontology. We distinguish two cases: (i) a concept with the same name as the original form of the phrase should be added into the ontology ('ADD') or (ii) a concept with a modified form of the phrase as its name should be added into the ontology ('ADD-m').

- (iii) No concept related to the phrase should be added to the ontology. This can happen because the phrase does not make sense in the domain ('No'), but also because it is a meaningful representation of a concept in a more general domain ('No-g'). For example, *electron transfer* is a general concept within the perspective in materials science, but should not necessarily be in an ontology for the nanotechnology domain.

A second interaction with the domain expert occurs in the **interpretation of topics**, which are outputs of ToPMine. The outcome can be one of the following.

- (i) Using the representative phrases in a topic, the domain expert labels the topic. Using this label as a phrase, we have the outcomes 'EXIST', 'EXIST-m', 'ADD', 'ADD-m', 'No-g' and 'No', as above. Furthermore, we add an outcome 'Q' (for query) when the label for the topic is too specific to add to the ontology, but could be defined using concepts in the ontologies and OWL constructs.

- (ii) Using a subset of representative phrases in a topic, the domain expert labels the subset. Using this label as a phrase, we have the outcomes 'EXIST', 'EXIST-m', 'ADD', 'ADD-m', 'No-g', 'No', and 'Q' as above. This can be done for different subsets.

Finally, the domain expert **interprets the lattice** which is generated based on the formal topical concept analysis.

- (i) Given the relationships in the lattice, as well as the connections of the topics and phrases to concepts in the ontology, new relationships between ontology concepts can be identified.

6.3 Summary

In this chapter, we have introduced an approach for ontology extension based on phrase-based topic modeling (ToPMine), formal topical concept analysis (FTCA) and domain expert validation. In the next chapter, we show how this approach contributes to extending ontologies in the nanotechnology domain and the materials design domain.

Chapter 7

Evaluation of ToPMine-FTCA

In this chapter, we present the related work (Section 7.1), and the evaluation (Section 7.2) of our approach, ToPMine-FTCA, as shown in Chapter 6. In the end we present a summary (Section 7.3).

7.1 Related work

As presented in Chapter 6, our approach (ToPMine-FTCA) mainly deals with concept discovery and concept hierarchy derivations. There are a number of relevant systems for extending ontologies. They are ASIUM [148], CRCTOL [149], OntoGain [150], OntoLearn [151] and Text2Onto [152]. ASIUM applies linguistics-based techniques including sentence parsing, syntactic structure analysis, and subcategorization frames and statistics-based clustering techniques to produce candidates to extend ontologies. CRCTOL implements both linguistics-based methods and relevance analysis. OntoGain extracts concepts by using linguistics-based techniques including part-of-speech tagging, sentence parsing, word sense disambiguation and statistics-based relevance analysis. OntoLearn generates concepts based on linguistics-based techniques including part-of-speech tagging and sentence parsing, as well as taking the concepts, glossary and hypernyms from WordNet into account. Text2Onto uses statistics-based co-occurrence analysis and linguistics-based techniques including part-of-speech tagging, sentence parsing, and syntactic structure analysis. For extracting concepts from the textual resource, Text2Onto implements four algorithms which are entropy-based, C-value/NC-value-based, relative term frequency-based, and term frequency and inverted document frequency (TF-IDF)-based respectively. We show the performance of these five systems in Table 7.1 according to [153]. Text2Onto is taken into account for a comparison with our ToPMine-FTCA. It is the only system that we could download and install. However, it is one of the most popular and well-known ontology learning systems and is therefore a good choice.

Table 7.1: Performance of ontology learning systems in different domains. (Precision is truncated.)

System	Domain	Precision
ASIUM	French journal Le Monde	0.86
CRCTOL	Patterns of Global Terrorism	0.92
OntoGain	Computer Science corpus	0.86
	Medical corpus	0.89
OntoLearn	Tourism	0.85
Text2Onto	Text from the paper [154]	0.61
	Patterns of Global Terrorism	0.74

7.2 Extending ontologies using ToPMine-FTCA

For the evaluation, we consider two cases facing the nanotechnology domain (presented in Section 7.2.1) and the materials design domain (presented in Section 7.2.2), respectively. The evaluation aims to answer the following research questions:

- **RQa**: How do the different outputs of the approach contribute to extending the domain ontologies?
- **RQb**: How does the approach compare with other methods?

7.2.1 Extending ontologies in the nanotechnology domain

For the nanotechnology domain, in [12] it is stated that there is a gap between data generation and shared data access. The domain lacks standards for collecting and systematically representing nanomaterial properties. In [13] stakeholder-identified technical and operational challenges for the integration of data in the nanotechnology domain are presented. The technical challenges mainly refer to (i) the use of different data formats, (ii) the use of different vocabularies, (iii) the lack of unique identifiers, and (iv) the use of different data conceptualization methods. In terms of operational challenges, they refer to (i) the fact that organizations have different levels of data quality and completeness, and (ii) the lack of understandable documentation.

In the rest of this section, we first introduce the two ontologies that we plan to extend using our approach, in Section 7.2.1.1. Then we introduce the unstructured data we have collected for extending ontologies in the nanotechnology domain, in Section 7.2.1.2. Then we show the experiments and the comparison with Text2Onto in Section 7.2.1.3 and Section 7.2.1.4, respectively.

7.2.1.1 Ontologies in the nanotechnology domain

The ontologies that we extend are the NanoParticle Ontology [91] and the eNanoMapper ontology [93]. Both ontologies are available via BioPortal.[1]

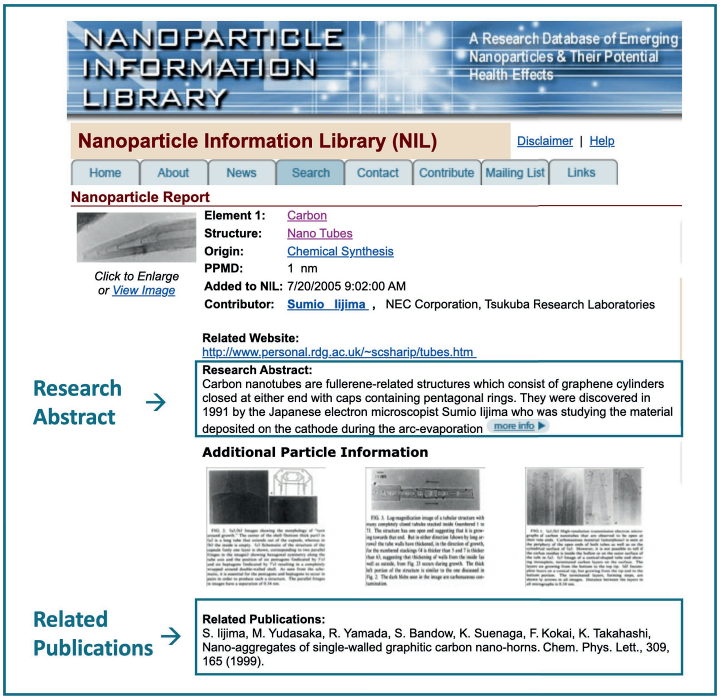
The NanoParticle Ontology [91] was created to support understanding biological properties of nanomaterials, searching for nanoparticle relevant data and designing nanoparticles. It builds on the Basic Formal Ontology (BFO)[2] [92] and the Chemical Entities of Biological Interest Ontology (ChEBI) [124] to represent basic knowledge regarding physical, chemical and functional features of nanotechnology used in cancer diagnosis and therapy. The ontology contains 1,904 concepts and 81 relations. The eNanoMapper ontology [93] aims to integrate a number of ontologies such as the NanoParticle Ontology to support assessing risks related to the use of nano materials. The ontology covers common vocabulary terms used in nano-safety research with a classification hierarchy (12,531 concepts) and other relations (4 relations).

7.2.1.2 Data collection

The corpus that we use is based on reports of nanoparticles from the Nanoparticle Information Library (NIL) [155], which is a research database of emerging nanoparticles. For each nanoparticle report, we take the text in the *'Research Abstract'* field as well as the abstracts (or only the title if there is no abstract) from the publications in the *'Related Publications'* field, as shown in Figure 7.1. The final corpus contains 117 abstracts from the collection according to the *'Research Abstract'* field and 510 publications from the collection

[1]`https://bioportal.bioontology.org/`
[2]`http://basic-formal-ontology.org/`

according to the *'Related Publications'* field, respectively. For these 510 publications, we include titles and abstracts in the final corpus. The title and abstract cover the basic content of an article. For research articles in the materials science domain, they will generally contain summaries of problems, experiments, simulations and computations. As the ontologies aim to represent basic knowledge in the domain, these parts of a research article often contain enough information for extraction of concepts. When using the full text, more proposals for concepts may be generated, but many of those will not be relevant. In other fields, it has been shown that the use of titles (and abstracts) may be a reasonable approach (e.g., [156]). Moreover, ToPMine is able to get valuable outputs based on corpuses consisting of titles and abstracts, as shown in [140].


NANOPARTICLE INFORMATION LIBRARY
A Research Database of Emerging Nanoparticles & Their Potential Health Effects
Disclaimer | Help
Home
About
News
Search
Contact
Contribute
Mailing List
Links
Click to Enlarge or View Image
more info
Research Abstract
Related Publications

Nanoparticle Information Library (NIL)

Nanoparticle Report

Element 1:	Carbon
Structure:	Nano Tubes
Origin:	Chemical Synthesis
PPMD:	1 nm
Added to NIL:	7/20/2005 9:02:00 AM
Contributor:	Sumio Iijima , NEC Corporation, Tsukuba Research Laboratories

Related Website:
http://www.personal.rdg.ac.uk/~scsharip/tubes.htm

Research Abstract:
Carbon nanotubes are fullerene-related structures which consist of graphene cylinders closed at either end with caps containing pentagonal rings. They were discovered in 1991 by the Japanese electron microscopist Sumio Iijima who was studying the material deposited on the cathode during the arc-evaporation

Additional Particle Information

Related Publications:
S. Iijima, M. Yudasaka, R. Yamada, S. Bandow, K. Suenaga, F. Kokai, K. Takahashi, Nano-aggregates of single-walled graphitic carbon nano-horns. Chem. Phys. Lett., 309, 165 (1999).

Figure 7.1: An example nanoparticle report in NIL.

7.2.1.3 Experiments

In Table 7.2, we show the detailed descriptions of different parameters used for ToPMine in our experiments for extending the NanoParticle Ontology, eNanoMapper ontology and MDO, respectively. These parameters can be classified into two groups, which are parameters for frequent phrases mining (i.e., *min_support*, *max_support_word*, *max_phrase_size*, and $alpha_1$) and topic modeling (i.e., *num_topics*, $alpha_2$, and *beta*). In our experiments for extending ontologies in the nanotechnology domain, we configure the phrases mining threshold ($alpha_1$) with two values (high and low), and the ToPMine with different numbers of requested topics (20, 30 and 40). The values of $alpha_2$ and *beta* as hyper-parameters are justified in [157]. Thus we have six experiments based on two values of $alpha_1$ and three values of *num_topics* over the data.

For the interpretation of the phrases, topics and lattice results, a domain expert worked together with two ontology engineering experts. In the first 2 hour session the three experts went through the phrases of all topics for one of the settings (low mining threshold, 40 topics) of the topic model approach. Each phrase was discussed regarding whether it is relevant for a nanotechnology ontology, a check was performed to determine whether concepts with the same or similar names already exist in the NanoParticle Ontology, and then decisions were made regarding a category of 'EXIST(-m)', 'ADD(-m)', or 'No(-g)' as well as which axioms may be necessary to add to the ontology. In addition to investigating the ontologies, in some cases terms were also checked via Wikipedia or research articles. In preparation for the second session, the ontology engineers prepared suggestions for the phrases for the other settings, based on the interpretation results of the first session and a search in the two ontologies. During the second session (4 hours) the phrases for all settings were interpreted and related to both ontologies, and the topics for one setting were interpreted. In the third (2 hour) session the remaining topics as well as the lattice results were interpreted.

After the interpretation of the phrases by the domain expert for each setting, all phrases interpreted with 'No' were removed from the phrase occurrence matrix. The updated matrix (with all 'EXIST(-m)', 'ADD(-m)' and 'No-g' phrases) were used as input for the formal topical concept analysis and a formal topical concept lattice was generated.

Validation of frequent phrases. In Table 7.3 we show the results regarding the interpretation of the phrases. In addition to the number of concepts in the 'EXIST(-m)', 'ADD(-m)', and 'No(-g)' categories, we also show the precision. The precision of the system is the ratio of the number of relevant

Table 7.2: The parameters of ToPMine and New ToPMine.

Parameter	Default	Description of the parameter	The value for extending NPO/eNM	The value for extending MDO
min_support	10	Minimum support threshold that each phrase must be less than during frequent pattern mining	10	10, 15, 20, 25, 30
max_phrase_size	40	Maximum allowed phrase size	10	10
$alpha_1$	4	Threshold for the significance score which must be satisfied when combining two words as a phrase	4 (low), 20 (high)	4
max_support_word	–	Maximum support threshold that each word in a phrase must be less than during frequent pattern mining	–	500, 1000, 3000 5000, 8000
num_topics	5	Number of requested topics that need to be extracted from the inputted documents	20, 30, 40	10, 20
$alpha_2$	4	A hyper-parameter representing the symmetric Dirichlet prior over document-topic distributions	50/*num_topics*	4
beta	0.01	A hyper-parameter representing the symmetric Dirichlet prior over topic-word distributions	0.01	0.01

Table 7.3: The result of interpreting phrases. The first column defines the case using the number of topics, low or high mining threshold, and ontology. The precision is truncated.

Setting	ADD	ADD-m	EXIST	EXIST-m	No-g	No	precision
20, low, NPO	32	4	26	19	16	9	0.91
20, low, eNM	29	3	24	25	14	12	0.88
30, low, NPO	30	4	26	18	16	9	0.91
30, low, eNM	28	3	24	26	12	11	0.89
40, low, NPO	32	4	26	15	16	10	0.90
40, low, eNM	29	3	24	22	14	12	0.88
20, high, NPO	9	1	14	7	4	0	1.00
20, high, eNM	8	2	12	10	3	0	1.00
30, high, NPO	8	2	14	8	0	1	0.96
30, high, eNM	7	1	12	10	0	1	0.96
40, high, NPO	9	2	14	12	4	4	0.91
40, high, eNM	9	2	12	14	2	4	0.90

For the meanings of 'ADD(-m)', 'EXIST(-m)' and 'No(-g)', see Section 6.2.3.
For 'ADD' and 'ADD-m', a new concept is defined in the ontology and one or more subsumption axioms are added.

proposed concepts to the number of proposed concepts. We defined a relevant proposed concept as a proposed concept that the domain expert recognizes as a relevant concept, whether it is in the ontology, or is more specific than concepts in the ontology, or if it could belong to a more general ontology. Therefore, the relevant proposed concepts are the ones that do not belong to the 'No' category. This conforms to what is relevant in the ontology learning setting.

We note that some phrases may contribute to the addition of multiple concepts and axioms. Furthermore, the low mining threshold settings generate the highest number of phrases (in total and per topic). Except for one 'No' phrase, all phrases generated by any of the high mining threshold settings are also generated by at least one (and usually all) low mining threshold settings. For the low mining threshold settings there are only small differences regarding the phrases that occur in topics. There are 29 phrases that are generated by all settings. Of these, 13 exist in the ontologies and relate, among others, to kinds of nanotubes, microscopy, spectroscopy, and various properties of nanoparticles. Furthermore, 7 exist in a modified form, e.g., *'core-shell nanoparticle'* for the phrase *'core shell'*. The remaining 9 should be added to the ontologies in the same or modified form. These relate to

Table 7.4: The number (and truncated percentage in parentheses) of topics that contribute to extending the ontologies. The first column defines the case using the number of topics, low or high mining threshold, and ontology.

Setting	contribute to ADD and ADD-m	contribute to EXIST and EXIST-m	contribute to No-g
20, low, NPO	18 (90.0%)	16 (80.0%)	6 (30.0%)
20, low, eNM	18 (90.0%)	16 (80.0%)	5 (40.0%)
20, high, NPO	11 (55.0%)	13 (65.0%)	3 (15.0%)
20, high, eNM	11 (55.0%)	13 (65.0%)	2 (10.0%)
30, low, NPO	19 (63.0%)	19 (63.0%)	11 (36.6%)
30, low, eNM	18 (60.0%)	20 (66.6%)	11 (36.6%)
30, high, NPO	10 (33.3%)	19 (63.3%)	3 (10.0%)
30, high, eNM	9 (30.0%)	20 (66.6%)	2 (6.6%)
40, low, NPO	22 (55.0%)	21 (52.5%)	12 (30.0%)
40, low, eNM	21 (52.5%)	23 (57.5%)	9 (22.5%)
40, high, NPO	13 (32.5%)	16 (40.0%)	4 (10.0%)
40, high, eNM	12 (30.0%)	18 (45.0%)	3 (7.5%)

properties (*‘resolution’*, *‘pore size’*, *‘band gap’*, *‘electrical conductivity’*, *‘crystallinity’*), a technique (*‘vapor deposition’*) and nano-objects (*‘mesoporous silica nanoparticle’*, *‘thin film’*). *‘Reverse micelle-synthesized quantum dot’* leads to the creation of a specific kind of quantum dots as well as a specific synthesis technique. Regarding the phrases that are only found by low mining threshold settings, they relate to different kinds of silicons, nanoparticles, properties and techniques, of which many should be added to the ontologies. There are, however, also several phrases that relate to more general concepts in the materials domain that should not necessarily be added to an ontology in the nanotechnology domain. In all settings, we find most ‘EXIST(-m)’ cases, which shows that the phrases are relevant with respect to the existing ontologies. Furthermore, we find many ‘ADD(-m)’ cases, which lead to new concepts and axioms. There are also some phrases that relate to more general concepts and some phrases that do not lead to anything meaningful in the context of extending the ontology. From Table 7.4 we note that the more topics the system generates, the lower the percentage of topics that contribute to ‘EXIST(-m)’ and ‘ADD(-m)’ categories.

Table 7.5: The result of interpreting topics. The first column defines the case using the number of topics, low or high mining threshold, and ontology. Note that some topics may be empty and some topics may require several concepts. The values in parentheses show the number of added concepts that are not found in the phrase interpretation phase.

Setting	ADD	ADD-m	EXIST	EXIST-m	No-g	Q	No	precision
20, low, both	3(1)	0	2	0	1	13	0	1.00
30, low, both	8(2)	0	4	0	1	13	0	1.00
40, low, both	16(1)	0	11	1	2	10	5	0.88
20, high, both	8(1)	0	3	2	0	7	0	1.00
30, high, both	3(2)	0	10	2	0	7	0	1.00
40, high, NPO	10(2)	0	10	3	2	3	2	0.93
40, high, eNM	10(2)	0	9	4	2	3	2	0.93

For the meanings of 'ADD(-m)', 'EXIST(-m)', 'No(-g)' and 'Q', see Section 6.2.3.
For 'ADD' and 'ADD-m', a new concept is defined in the ontology and one or more subsumption axioms are added.

Validation of topics. In Table 7.5 we show the results regarding the interpretation of the topics. We note that the high mining threshold settings generate the most concepts to add to the ontologies. In each setting there are one or two concepts that are not found during the interpretation of the phrases (e.g., *'high resolution experiment'*, *'water soluble reverse micelle systems'*, *'core-shell semiconductors'*). All 'EXIST(-m)' concepts are also found during the interpretation of the phrases. The 'No-g' category consists of previously identified phrases or specializations of those. Furthermore, many of the topics are very specific and it is decided they should not be added to the ontology, but queries (or complex concepts) using concepts in the ontologies and OWL constructs can be constructed. We also observe that the results for the two ontologies are almost the same, which may be because the topic labels are (much) more specific than the phrase labels and the ontologies do not model concepts at the lowest levels of specificity.

Validation of topical lattices. In the final step we generate lattices for all settings. As an example, a part of the lattice for the case of 40 requested topics with a low mining threshold is shown in Figure 7.2. Nodes that contain one topic/one phrase with the bottom node as their child and the top node as their parent are not shown. These have been dealt with in the phrase interpretation step and as there are no connections to other nodes (except top and bottom), no additional information can be gained for those nodes.

Table 7.6: The result of interpreting lattice nodes. The first column defines the case using the number of topics, low or high mining threshold, and ontology. The values in parentheses show the number of added concepts that are not found in the phrase or topic interpretation phases.

Setting	ADD	ADD-m	EXIST	EXIST-m	No-g	Q	No	precision
20, low, both	1(0)	0	1	0	2	0	0	1.00
30, low, NPO	4(2)	0	3	0	1	0	0	1.00
30, low, eNM	3(2)	0	4	0	1	0	0	1.00
40, low, both	3(0)	0	1	0	0	0	0	1.00
20, high, both	0(0)	0	1	0	1	1	0	1.00
30, high, both	1(1)	0	1	0	0	0	0	1.00
40, high, both	0(0)	0	0	0	0	0	0	1.00

For the meanings of 'ADD(-m)', 'EXIST(-m)', 'No(-g)' and 'Q', see Section 6.2.3.
For 'ADD', a new concept is defined in the ontology and one or more subsumption axioms are added.

The lattices are used in the following ways. First, the domain expert labels the nodes based on the phrases connected to the nodes. These may be the extents or subsets of the extents of topics. The results are given in Table 7.6. Some new concepts are found that are more general than concepts related to topics (e.g., *'core-shell cdse nanoparticles'*), but in general, little additional information is found.

Secondly, the domain expert labels the nodes based on the phrases connected to the nodes and their descendants. As a node contains fewer phrases than all of its ancestors, labeling may lead to the definition of a new concept that is a super-concept of the concepts related to the ancestor topics (and relevant axioms). As, according to the topic interpretation step, many topics are very specific, this approach may provide a way to decide on the appropriate level of specificity for concepts to add to the ontology. In our experiments, however, the lattices are very flat and the nodes with empty intent contained only one phrase, thus they do not lead to additional concepts.

Thirdly, the domain expert uses the lattice as a visualization tool to check the original topic interpretation. According to the domain expert, the use of the lattice provides significant help in interpreting the topics. As it groups phrases that different topics have in common and distinguishes phrases that are specific for certain topics, the structure of complex concepts (based on other concepts) is clarified. This results in a better organization and visualization of the topics and their underlying notions. For instance, for a topic with phrases *'particle size'*, *'quantum dot'*, and *'gold nanoparticle'*, the phrase *'particle size'* is shared with another topic. By removing *'particle size'* from the phrase list of the topic, it is easier to see that the topic is a combination of *'particle size'* and a notion of *'quantum dots of gold nanoparticles'*.

Summary of validations. For our experiments we have currently used a small number of resources, i.e., circa 600 abstracts and less than 10 hours for each of the three experts. Even with these limited resources our approach finds 35 and 32 new concepts for the NanoParticle Ontology and the eNanoMapper ontology, respectively, as shown in Table 7.7, as well as 42 and 37 new axioms, respectively, as shown in Table 7.8. In addition to the new concepts and new axioms, also other concepts are influenced. Indeed, for a new axiom A is-a B, the sub-concepts of A receive B and all its super-concepts as its super-concepts (and thus inherit their properties), and all super-concepts of B receive A and its sub-concepts as sub-concepts (and thus all instances of these concepts are also instances of B and its super-concepts). In this experiment, 72 concepts from NanoParticle Ontology are influenced by the new axioms. Therefore, the quality of semantically-enabled applications is improved whenever one of the 35 new or 72 influenced concepts is used. For the eNanoMapper ontology the number of existing concepts influenced by adding new axioms is 37. In general, if domain and range are used for the definition of relations in the ontologies, even more concepts would be influenced. Thus, adding these axioms improves the quality of the ontologies and the semantically-enabled applications that use these ontologies. It is clear that the effort of extending the ontologies is worthwhile.

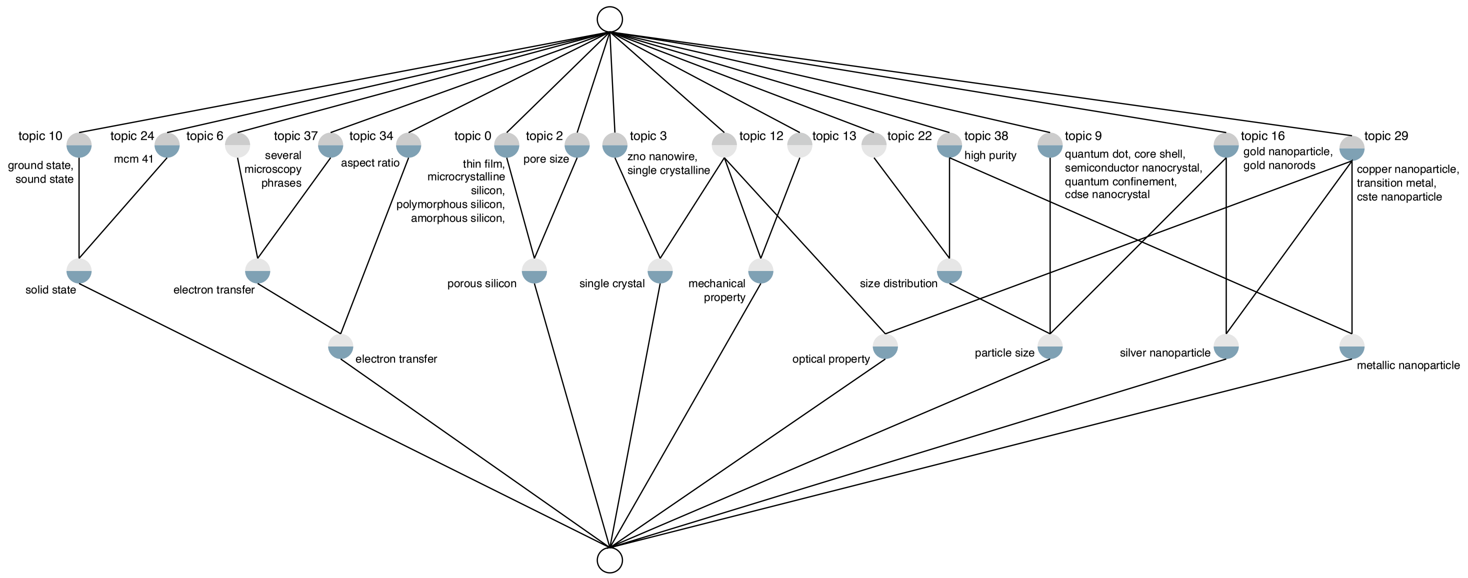

topic 10
ground state, sound state
topic 24
mcm 41
topic 6
topic 37
several microscopy phrases
topic 34
aspect ratio
topic 0
thin film, microcrystalline silicon, polymorphous silicon, amorphous silicon,
topic 2
pore size
topic 3
zno nanowire, single crystalline
topic 12
topic 13
topic 22
topic 38
high purity
topic 9
quantum dot, core shell, semiconductor nanocrystal, quantum confinement, cdse nanocrystal
topic 16
gold nanoparticle, gold nanorods
topic 29
copper nanoparticle, transition metal, cste nanoparticle
solid state
electron transfer
porous silicon
single crystal
mechanical property
size distribution
electron transfer
optical property
particle size
silver nanoparticle
metallic nanoparticle

Figure 7.2: Part of the lattice for the 40 topics and low mining threshold setting. Nodes that contain one topic/one phrase and have as child the bottom node and as parent the top node are not shown.

Table 7.7: New concepts for the NanoParticle Ontology and the eNanoMapper ontology.

Concept	NanoParticle	eNanoMapper
amorphous silicon	✓	
band gap	✓	
Barium Titanate	✓	✓
block copolymer	✓	✓
copolymer	✓	✓
polymer		✓
CdSe nanocrystal	✓	✓
CdTe nanoparticle	✓	✓
copper nanoparticle	✓	
conductivity	✓	✓
electrical	✓	✓
gold nanorod	✓	✓
growth mechanism	✓	✓
resolution	✓	✓
layer by layer growth	✓	✓
liquid solid	✓	
pressure	✓	
MCM 41	✓	✓
mechanical property	✓	✓
viscosity		✓
melt spin	✓	✓
mesoporous silica nanoparticle	✓	✓
mesoporous silica nanosphere	✓	✓
microcrystalline silicon	✓	✓
optical property		✓
polymorphous silicon	✓	✓
pore size	✓	
porous silicon	✓	✓
quantum confinement	✓	✓
reverse micelle-type quantum dot	✓	✓
semiconductor nanocrystal	✓	✓
nanocrystal	✓	✓
silicon thin film	✓	✓
thin film	✓	✓
crystallinity	✓	✓
thermal conductivity	✓	✓
tunnel spectroscopy	✓	✓
ZnO nanowire	✓	✓
	35	**32**

Table 7.8: New axioms for the NanoParticle Ontology and the eNanoMapper ontology.

Axiom	NanoParticle	eNanoMapper
amorphous silicon is a silicon	✓	
band gap is a quality	✓	
Barium Titanate is an inorganic compound or molecule	✓	
Barium Titanate is a chemical substance		✓
block copolymer is a copolymer	✓	✓
copolymer is a polymer	✓	✓
polymer is an organic material		✓
CdSe nanocrystal is a nanocrystal	✓	✓
CdTe nanoparticle is a nanoparticle	✓	✓
copper nanoparticle is a metal nanoparticle	✓	
conductivity is an independent general individual quality	✓	
conductivity is a quality		✓
electrical conductivity is a conductivity	✓	✓
gold nanorod is a nanorod	✓	✓
growth mechanism is a process	✓	✓
resolution is an independent general individual quality	✓	
resolution is a quality		✓
layer by layer growth is a mechanism process	✓	✓
liquid solid is a liquid solid interface	✓	
pressure is an independent general individual quality	✓	
MCM 41 is a mesoporous silica nanoparticle	✓	✓
mechanical property is a realizable entity	✓	
mechanical property is a quality		✓
viscosity is a mechanical property	✓	✓
melt spin is a technique	✓	✓
mesoporous silica nanoparticle is a nanoparticle	✓	✓
mesoporous silica nanosphere is a nanosphere	✓	✓
microcrystalline silicon is a silicon	✓	
microcrystalline silicon is a chemical substance		✓
nanotube array has part nanotube	✓	✓
optical property is a property		✓
polymorphous silicon is a silicon	✓	
polymorphous silicon is a chemical substance		✓
pore size is a nanoparticle property	✓	
porous silicon is a silicon	✓	
porous silicon is a chemical substance		✓
raman scatter is a synonym of raman spectroscopy	✓	✓
quantum confinement	✓	✓
reverse micelle-type quantum dot is a quantum dot	✓	✓
semiconductor nanocrystal is a semiconductor and is a nanocrystal	✓	✓
nanocrystal is a nano-object and is a crystal	✓	✓
silicon thin film is a thin film	✓	✓
thin film is a fiat material part and one-dimensional nano-object	✓	✓
crystallinity is an independent general individual quality	✓	
crystallinity is a quality		✓
transition metal is a synonym of transition element	✓	
thermal conductivity is a conductivity	✓	✓
tunnel spectroscopy is a spectroscopy	✓	✓
scanning tunneling spectroscopy is same as tunnel spectroscopy	✓	✓
chemical vapor disposition is a vapor disposition	✓	✓
physical vapor disposition is a vapor disposition	✓	✓
ZnO nanowire is a nanowire	✓	✓
	42	**37**

7.2.1.4 Comparison with Text2Onto

In this experiment, we use Text2Onto on the same corpus as in the experiment for our approach. We apply Text2Onto to our corpus with default settings for its four algorithms. For each of the settings, Text2Onto returns thousands of candidates ranked by relevance. Instead of using the complete ranked lists of thousands of proposed concepts, we decided to investigate the results of the sub-lists containing the 100, 200, 300 and 400 top candidates in the lists, respectively. The results are shown in Table 7.9. The entropy-based and C-value/NC-value-based methods return exactly the same results. For the relative term frequency (RTF)-based method the 160 highest ranked proposed concepts are the same as the 160 highest ranked proposed concepts for the entropy-based and C-value/NC-value-based methods. The precision for the entropy-based and C-value/NC-value-based methods is the highest for each fixed number of proposed concepts, closely followed by the relative term frequency-based method. The TF-IDF-based method has the lowest precision. However, the TF-IDF-based method finds the largest number of relevant new concepts (‘ADD(-m)’). Furthermore, the precision decreases and the number of relevant new concepts increases for all algorithms when we take larger sub-lists of top elements.

In Table 7.10, we show the results for Text2Onto when all algorithms are used together for the different sub-lists of top elements and compare it to our method. To answer **RQb**, in Table 7.11 we show all the new concepts found by our method and Text2Onto for NanoParticle Ontology. 14 concepts are found by both methods. Additionally, our method finds 21 new concepts that are not found by Text2Onto, while Text2Onto finds 28 new concepts that are not found by our method. The two methods seem, therefore, to be complementary.

Table 7.9: The results of Text2Onto with different algorithms and different numbers of returned candidates. (Precision is truncated.)

No.	Algorithm	ADD	ADD-m	EXIST	EXIST-m	No-g	No	precision
100	Entropy	5	0	39	19	4	33	0.67
	C-value/NC-value	5	0	39	19	4	33	0.67
	RTF	5	0	39	20	4	32	0.68
	TF-IDF	17	0	22	12	6	43	0.57
200	Entropy	7	1	63	43	8	79	0.60
	C-value/NC-value	7	1	63	43	7	79	0.60
	RTF	7	1	63	42	8	79	0.60
	TF-IDF	24	1	38	19	19	99	0.50
300	Entropy	12	1	80	52	16	139	0.53
	C-value/NC-value	12	1	80	52	16	139	0.53
	RTF	13	1	78	52	16	140	0.53
	TF-IDF	28	1	58	36	29	148	0.50
400	Entropy	18	1	98	62	20	199	0.50
	C-value/NC-value	18	1	98	62	20	199	0.50
	RTF	19	1	100	61	20	199	0.50
	TF-IDF	36	1	70	44	38	211	0.47

Table 7.10: The results for Text2Onto using all algorithms per setting and ToPMine-FTCA for extending the NanoParticle Ontology. (Precision is truncated.)

Setting	ADD	ADD-m	EXIST	EXIST-m	No-g	No	precision
Text2Onto-100	20	0	51	27	11	71	0.60
Text2Onto-200	29	1	84	55	26	164	0.54
Text2Onto-300	39	1	118	78	44	266	0.51
Text2Onto-400	41	1	120	73	47	313	0.47
ToPMine-FTCA	32	3	25	18	14	22	0.80

Table 7.11: New concepts found by ToPMine-FTCA and Text2Onto for the NanoParticle Ontology.

Concept	Approach	Concept	Approach
amorphous silicon	$\checkmark^{tf}$	intensity	$\checkmark^{t2o}$
crystallinity	$\checkmark^{tf}$	pressure	$\checkmark^{t2o}$
CdSe nanocrystal	$\checkmark^{tf}$	melting	$\checkmark^{t2o}$
CdTe nanoparticle	$\checkmark^{tf}$	nano colloid	$\checkmark^{t2o}$
electrical conductivity	$\checkmark^{tf}$	nano composite	$\checkmark^{t2o}$
resolution	$\checkmark^{tf}$	nano crystalline silicon particle	$\checkmark^{t2o}$
layer by layer growth	$\checkmark^{tf}$	nanogrid	$\checkmark^{t2o}$
liquid solid	$\checkmark^{tf}$	nano ribbon	$\checkmark^{t2o}$
MCM 41	$\checkmark^{tf}$	nanowire array	$\checkmark^{t2o}$
mechanical property	$\checkmark^{tf}$	oxidation	$\checkmark^{t2o}$
melt spin	$\checkmark^{tf}$	photo activity	$\checkmark^{t2o}$
mesoporous silica nanoparticle	$\checkmark^{tf}$	polyelectrolyte	$\checkmark^{t2o}$
mesoporous silica nanosphere	$\checkmark^{tf}$	silica nanosphere	$\checkmark^{t2o}$
polymorphous silicon	$\checkmark^{tf}$	silicon nanowire	$\checkmark^{t2o}$
porous silicon	$\checkmark^{tf}$	silicon nanowire array	$\checkmark^{t2o}$
reverse micelle-type quantum dot	$\checkmark^{tf}$	superlattice nanowire	$\checkmark^{t2o}$
silicon thin film	$\checkmark^{tf}$	titanium nanotube	$\checkmark^{t2o}$
thin film	$\checkmark^{tf}$	band gap	$\checkmark^{both}$
thermal conductivity	$\checkmark^{tf}$	barium titanate	$\checkmark^{both}$
tunnel spectroscopy	$\checkmark^{tf}$	block copolymer	$\checkmark^{both}$
ZnO nanowire	$\checkmark^{tf}$	copolymer	$\checkmark^{both}$
acid group	$\checkmark^{t2o}$	copper nanoparticle	$\checkmark^{both}$
activation energy	$\checkmark^{t2o}$	conductivity	$\checkmark^{both}$
barium titanate nanowire	$\checkmark^{t2o}$	gold nanorod	$\checkmark^{both}$
boron nanowire	$\checkmark^{t2o}$	growth mechanism	$\checkmark^{both}$
catalyst	$\checkmark^{t2o}$	microcrystalline silicon	$\checkmark^{both}$
cluster	$\checkmark^{t2o}$	nanocrystal	$\checkmark^{both}$
crystallite	$\checkmark^{t2o}$	nanotube array	$\checkmark^{both}$
diblock copolymer	$\checkmark^{t2o}$	pore size	$\checkmark^{both}$
esterification	$\checkmark^{t2o}$	quantum confinement	$\checkmark^{both}$
ethylene oxide	$\checkmark^{t2o}$	semiconductor nanocrystal	$\checkmark^{both}$

$\checkmark^{both}$ represents that the concept is found by both ToPMine-FTCA and Text2Onto.
$\checkmark^{tf}$ represents that the concept is found only by ToPMine-FTCA, while $\checkmark^{t2o}$ represents that the concept is found only by Text2Onto.

7.2.2 Extending Materials Design Ontology

Although the Materials Design Ontology (MDO) presented in Chapter 5 fills a gap in the materials design domain in terms of covering domain knowledge, there is still room for modeling additional relevant concepts and relationships. In this section, we present how we apply our ToPMine-FTCA approach to extending MDO. As we show in Section 7.2.1.2, for extending ontologies in the nanotechnology domain, we make use of the Nanoparticle Information Library, which is a research database that gathers relevant works about nanoparticles. However, there is no similar corpus or database gathering related research papers that we can use for mining the unstructured data in the materials design field. This is because materials design is a general process that can cover, for instance, structural information or calculation information, as opposed to nanoparticles, which represent a specific kind of materials in terms of the nanotechnology domain Therefore, we make an extra effort in terms of collecting the corpus and applying more techniques for processing the collected corpus.

During the data collection process, we use MDO as a seed for querying journal databases. We use two journals in the field of materials design, which are NPJ Computational Materials[3] and Computational Materials Science.[4] We use the 37 concepts of MDO as search phrases in the two journals to find relevant articles and then retrieve the titles and abstracts of the returned articles. Upon completion of this process, the corpus contains titles and abstracts from 403 articles of NPJ Computational Materials and 8,193 from Computational Materials Science. When using ToPMine-FTCA on the corpus, we add a preprocessing step when preparing input for ToPMine and a selection step on words before performing frequent phrase mining. The purpose of the former step is to provide a more precise corpus to ToPMine, since the corpus may be more general than the one we use for extending ontologies in the nanotechnology domain. The purpose of the latter step is to generate more precise frequent phrases.

7.2.2.1 Preprocessing for ToPMine

In the preprocessing step, characters are set to lower case and punctuation is removed. We also remove words with a word length of either one or two.

[3]`https://www.sciencedirect.com/journal/computational-materials-science`
[4]`https://www.nature.com/npjcompumats/`

Such words are also general stopwords. After preprocessing there are 21,548 distinct words, which together occur 808,862 times. An overview of the frequency of the words is presented in Table 7.12. Most of the words (72.27%) occur less than 10 times, while there are 17 words that occur more than 3000 times. These are *'based'*, *'properties'*, *'method'*, *'calculations'*, *'phase'*, *'materials'*, *'study'*, *'structure'*, *'temperature'*, *'density'*, *'results'*, *'energy'*, *'electronic'*, *'model'*, *'molecular'*, *'simulations'*, and *'surface'*.

Table 7.12: The distribution of word frequency after preprocessing.

Frequency	Percentage
less than 10	72.27%
10-30	13.25%
31-100	7.76%
101-500	5.25%
501-1000	0.83%
1001-2000	0.44%
2001-3000	0.12%
More than 3000	0.08%

7.2.2.2 Selecting frequent phrases

Given a minimum support threshold *min_support* in ToPMine, the phrases that occur at least *min_support* times can be *frequent phrases*. ToPMine also generates frequent phrases of a length up to a maximum length that is given as an input parameter (*max_phrase_size* as shown in Table 7.2). Furthermore, ToPMine does not generate all frequent phrases, rather it uses a method based on partitioning documents and using a significance score to decide which words are likely to belong together, in order to produce high-quality frequent phrases [140]. The second column of Table 7.13 shows the number of frequent phrases that ToPMine generates for different values of *min_support*. The higher the *min_support*, the fewer frequent phrases are generated.

In addition, we also define a maximum support threshold *max_support_word*, and those words that occur more than *max_support_word* times are removed. That is to say, we do not take such words into account when composing phrases in ToPMine. These words are usually very general terms that are not interesting for an ontology or that would not be interesting for a domain ontology, though they might be

Table 7.13: Number of frequent phrases for *min_support* as 10, 15, 20, 25 and 30 respectively, and three different versions of the ToPMine algorithm.

min_support	original TopMine	New ToPMine without stemming	New ToPMine with stemming
10	6,901	6,478	5,452
15	3,826	3,578	3,022
20	2,542	2,402	2,046
25	1,816	1,722	1,477
30	1,375	1,298	1,119

Table 7.14: Number of frequent phrases for *min_support* as 10 and for *max_support_word* as 500, 1000, 3000, 5000, and 8000, respectively for two different versions of the ToPMine algorithm.

max_support_word	New ToPMine without stemming	New ToPMine with stemming
8,000	6,478	5,452
5,000	5,947	5,023
3,000	4,692	4,090
1,000	1,878	1,692
500	932	866

interesting for an upper ontology. We do note, however, that some of these words could be useful, such as *'method'*, *'electronic'*, *'model'*, and *'molecular'*. In the remainder of this chapter we refer to the algorithm that adds *max_support_word* as well as the preprocessing step as **New ToPMine**. The second column in Table 7.14 shows how *max_support_word* influences the number of generated frequent phrases with a constant *min_support* of 10. The higher the value of *max_support_word*, the more frequent phrases are generated. Since that no word occurs more than 8000 times in our corpus, setting *max_support_word* to 8000 allows all words (or, in other words, *max_support_word* is not used).

Another way to look at the influence of *min_support* and *max_support_word* is to compare how many of the frequent phrases are the same and how many are different for different settings. In Figure 7.3 we show this comparison of different settings to the base setting where *min_support* is 10 and *max_support_word* is 8000 (i.e., *max_support_word*

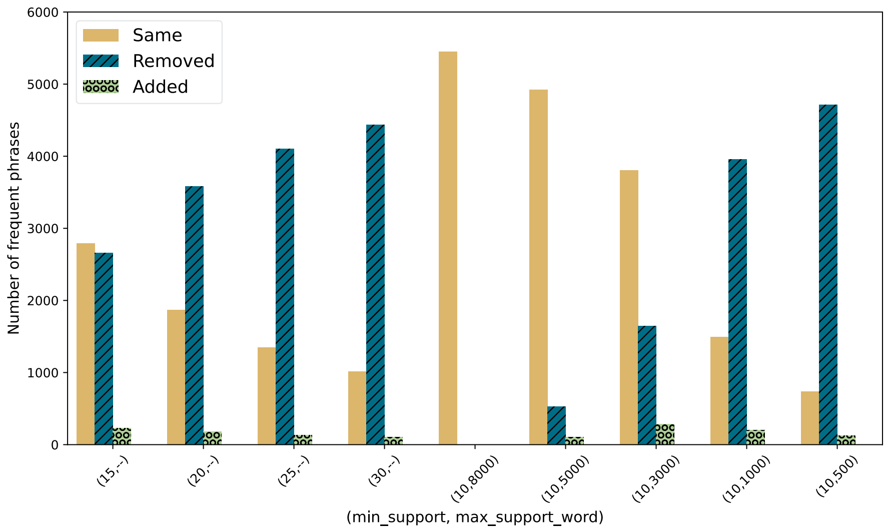


Figure 7.3: Comparison of the frequent phrases of New ToPMine algorithm with *min_support* as 10 (and *max_support_word* as 8000) to settings with *min_support* as 15, 20, 25 and 30, respectively, and settings with *min_support* as 10 and *max_support_word* as 500, 1000, 3000, 5000, respectively.

is not used), which is shown in the middle of the figure. The ‘Same’ bars show how many generated phrases occur both in the base setting and the compared setting. The ‘Removed’ bars show how many frequent phrases occur in the base setting, but not in the compared setting. For the cases where we change *min_support*, these would be phrases that are frequent phrases for *min_support* as 10, but not for the higher *min_support* value in the compared setting. For example, *‘computational screening’* is removed for *min_support* 15. For the cases where we change the *max_support_word*, these would be phrases with words that occur more often than the *max_support_word* in the compared setting. For instance, *‘sheet metal forming’* contains the word *‘metal’*, which has a frequency of 3,457 and would thus be removed for *max_support_word* as 1000. The ‘Added’ bars show which frequent phrases occur newly in the compared settings. This happens, as previously stated, because ToPMine does not generate all frequent phrases, but instead focuses on high-quality frequent phrases. As an example, *‘exchange correlation potential’* appears at least 10 times and less than 30 times and *‘exchange correlation’* appears at least 30 times. Both are frequent phrases for *min_support* as 10. However, ToPMine does not generate *‘exchange* *correlation'* for *min_support* 10, but it does generate *'exchange correlation potential'*. For *min_support* as 30, *'exchange correlation potential'* is not a frequent phrase, while *'exchange correlation'* is, and ToPMine does generate *'exchange correlation'* as a frequent phrase.

We also investigate using stemming on the frequent phrases. As an example, the phrases *'molecular dynamics simulations'*, *'molecular dynamics simulation'*, *'molecular dynamic simulations'* and *'molecular dynamic simulation'* have the same stem *'molecular dynam simul'*. Stemming allows for removing redundant phrases and thus reduces the work of the domain expert. The influence on the number of generated phrases can be seen by comparing the last two columns in Tables 7.13 and 7.14. A disadvantage is that in some cases possible concept candidates may be removed. To alleviate this problem we show the domain expert for each of the stemmed frequent phrases the list of corresponding original phrases. This also helps the domain expert to choose terms to be added to the ontology.

In Table 7.15, we show the candidate concepts based on the validation by a domain expert of the frequent phrases from the experiment with *min_support* as 30 and *max_support_word* as 500. In total, 88 candidate concepts are suggested based on 81 out of 131 frequent phrases generated by the experiment. Some candidate concepts can be added into MDO as sub-concepts of existing concepts. For instance, *'Linearized Augmented Plane Wave Method'* is a subconcept of *'Density Functional Theory Method'*. Some candidate concepts are relevant to the materials design domain but may be not interesting for data access or data integration over materials design databases. For instance, *'Covalent Bond'* is a bonding type that can be used to describe materials structures.

7.2.2.3 Validating topics

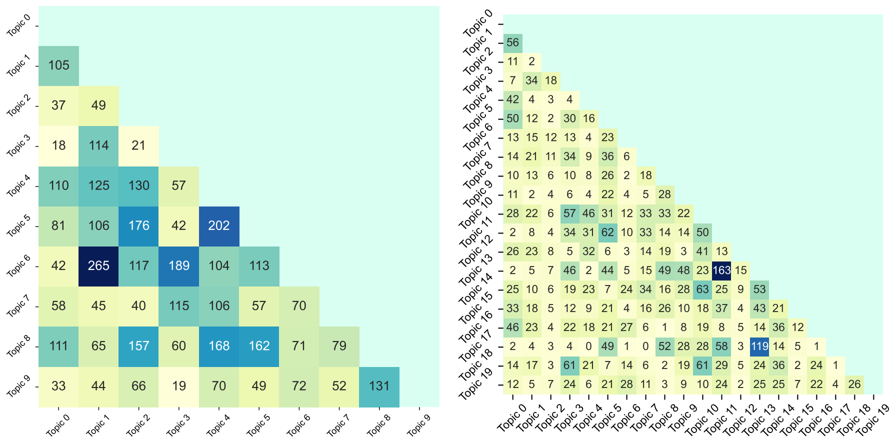
The number of topics (*num_topic*) is an input parameter to ToPMine. Each topic contains a set of phrases and these sets do not have to be disjoint. For instance, Figure 7.4 shows the overlap of phrases between topics for different settings of input parameters. In general, when we increase the number of topics, the number of frequent phrases in each topic decreases and the overlap between topics decreases as well.

The domain expert validated these topics and, if possible, labeled them to generate concepts for the ontology. In Table 7.16, we show the domain ex-

Table 7.15: Candidate concepts based on domain expert validation on the experiment with *min_support* as 30 and *max_support_word* as 500.

Iron	Charpy Impact Test
Zigzag	Ductile Transition
Armchair	Real Space Methods
Kohn-Sham	Solute Segregation
Rock Salt	Stone-wales Defect
Unit Cell	Absorption Spectrum
Core Shell	Body Centered Cubic
Rare Earth	Cohesive Zone Model
Slip Plane	Face Centered Cubic
Domain Wall	Hall-Petch Relation
Quantum Dot	Kinematic Hardening
Reuss Model	Mixed Mode Fracture
Zinc Blende	Rock Salt Structure
Cement Paste	Van der Waals Force
Porous Media	Alkaline Earth Metal
Power Factor	Coarse Grained Model
Valence Band	Homo-lumo Energy Gap
Voight Model	Quasi-harmonic Model
Anatase (TiO_2)	Anomalous Hall Effect
Boron Nitride	Carbon Nanotube (cnt)
Contact Angle	Additive Manufacturing
Covalent Bond	Cahn–Hilliard Equation
Fatigue Limit	Double Walled Nanotube
Lennard Jones	Spinodal Decomposition
Brillouin Zone	Hexagonal Boron Nitride
Edurance Limit	Microstructural Features
Stacking Fault	Spontaneous Polarization
Sound Velocity	Muffin-tin Orbital Method
Conduction Band	Austenitic Stainless Steel
Glass Formation	Brittle-Ductile Transition
Cauchy-Born Rule	Directional Solidification
Domain Switching	Quasi-harmonic Debye Model
Fiber Reinforced	Crystallographic Orientation
Half Metallicity	Functionally Graded Material
Nearest Neighbor	Hexagonal Close Packed (hcp)
Refractive Index	Rutile Titanium Dioxide (TiO_2)
Stainless Steels	Modified Embedded Atom Method
Vapor Deposition	Projector Augmented Wave Method
Vickers Hardness	Muffin-tin Orbital Approximation
X-ray diffration	Linearized Augmented Plane Wave Method
Dispersion Curves	Asymmetric Tilt Grain Boundary Structure
Vibrational Modes	Symmetric Tilt Grain Boundary Structure
Absorption Spectra	Modified Becke-Johnson Exchange-Correlation Functional
Brittle Transition	Perdew-Burke-Ernzerhof (PBE) Exchange-Correlation Functional

pert's validation of 10 topics generated by the New ToPMine with stemming, *min_support* of 30 and *max_support_word* of 500. Among these topics, there are two topics (topics 0 and 9) that are interpreted with multiples labels, i.e., the domain expert divides the topic in different parts. The other topics received one label. Further, representative phrases are given for each topic. The labels and the representative phrases can all lead to new concepts.



(a) *min_support as 10, num_topic as 10.* **(b)** *min_support as 10, num_topic as 20.*

Figure 7.4: Number of common phrases between pairs of topics.

Table 7.16: Topic labeling based on domain expert validation on the experiment with *min_support* as 30 and *max_support_word* as 500 (Up to five representative phrases are selected for each label.).

No.	Topic labels	Representative Phrases
0	Computational Method Categories	Linearized Augmented Plane Wave Method, Hartree-Fock Method, Kohn-Sham, Perdew-Burke-Ernzerhof (PBE) Exchange-Correlation Functional, Modified Becke-Johnson Exchange-Correlation Functional,
	Materials Properties and Features	Absorption Spectrum, Refractive Index, Homo-lumo Energy Gap, Alkaline Earth Metal, Dispersion Curves
	Electronic Structure Features	Conduction Band, Valence Band
	Materials Categorizations	Half Metallicity, Rare Earth
	Experimental Method Categories	X-ray Diffraction
	Specific Materials	Zinc Blende
	Applications	Optoelectronic Devices
1	Hardness-related Materials Concepts	Quasi-harmonic Debye Model, Quasi-harmonic Model, Rock Salt, Sound Velocity, Zinc Blende
2	Materials Strength-related Concepts	Stacking Fault, Van der Waals Force, Tension Compression, Uniaxial Tension, Symmetric Tilt Grain Boundary Structure
3	Materials Fatigue/Fracture-related Concepts	Functionally Graded Material, Fiber Reinforced, Cohesive Zone Model, Unit Cell, Cement Paste
4	Materials Synthesis Concepts	Additive Manufacturing, Vapor Deposition, Directional Solidification, Microstructural Features, Crystallographic Orientations
5	Battery-related Materials Concepts	Ion Batteries, Anatase (TiO_2), Lithium Ion Batteries, Rutile Titanium Dioxide (TiO_2), Boron Nitride
6	Materials Structural Categorizations	Face Centered Cubic, Body Centered Cubic, Coarse Grained Model, Hexagonal Close Packed (hcp), Iron
7	Nanotube-related Concepts	Armchair, Boron Nitride, Hexagonal Boron Nitride, Carbon Nanotube (cnt), Cross Section
8	Artificial Intelligence-Methods (NO)	Artificial Neural, Neural Networks, Open Source, Degrees Freedom, Artificial Neural Networks
9	Materials Concepts for Solar-cells	Solar Cells, Quantum Dots, Domain Wall, Power Factor, Electric Fields
	Materials Magnetism Concepts	Domain Switching, Anomalous Hall Effect
	Materials Polarization Concepts	Spontaneous Polarization

7.3 Summary

In this chapter, we have presented our evaluation of using ToPMine-FTCA to extend ontologies in the nanotechnology domain and the materials design domain. In the former case, with the help of a well-organized repository of relevant works for constructing the corpus, both our approach and Text2Onto produce reasonable candidates for extending the NanoParticle Ontology and the eNanoMapper ontology. In the latter case, we have shown the efforts we make for producing more precise candidates for domain experts to validate, in the situation that there is no organized repository of relevant works for constructing the corpus. Nevertheless, our approach produces relevant candidates. Since our Materials Design Ontology is relatively small, such candidates can be of interest with regard to ontologies for other specific domains.

CHAPTER

8

Evaluation of the GraphQL-based framework

In this chapter, we present an evaluation of the framework shown in Chapter 3. We consider a real case application scenario in the materials design domain in Section 8.1, and a synthetic benchmark scenario based on the Linköping GraphQL Benchmark (LinGBM)[1] in Section 8.2. Finally, the chapter ends with a summary in Section 8.3.

The evaluation aims to answer the following research questions:

- **RQa**: Can the generated GraphQL server provide integrated access to heterogeneous data sources?
- **RQb**: Can a GraphQL server generated based on the ontology answer queries that correspond to competency questions of the ontology?
- **RQc**: How does the generated GraphQL server compare to other Ontology-Based Data Access (OBDA) systems and other GraphQL-based systems in terms of query performance and its behavior for increasing dataset sizes?

We performed all experiments on a server machine with Intel Xeon Gold 6130 @ 2.10GHz CPUs. The machine runs a 64-bit CentOS Linux 7 (Core) operating system. We reserved 8 CPU cores and 4GB memory for the experiments.

[1]`https://github.com/LiUGraphQL/LinGBM`

8.1 Real case evaluation

In the real case evaluation, we focus on a use case in the materials design domain where the task is data integration over two data sources, Materials Project [15] and OQMD [158]. We compare our tool, OBG-gen (Ontology-Based GraphQL Server Generation) in two versions (OBG-gen-rdb and OBG-gen-mix) with three systems: morph-rdb [70], HyperGraphQL [72], and UltraGraphQL [74]. Morph-rdb is an OBDA tool that can access a relational database as a data source by translating SPARQL queries into SQL queries based on R2RML mappings. HyperGraphQL and its extension UltraGraphQL are GraphQL interfaces that can query Linked Data that may be provided by local RDF files and remote SPARQL endpoints. The semantic mappings (for all the systems) are based on the Materials Design Ontology presented in Chapter 5. OBG-gen generates the GraphQL schema based on MDO. The entire GraphQL schema is shown in Appendix B.1. UltraGraphQL and HyperGraphQL use a modified version of the generated schema since they require directive definitions to specify the correspondences between query entries and the data.

8.1.1 Data

The data from the Materials Project and OQMD represents five different types of real-world entities (*Calculation*, *Structure*, *Composition*, *Band Gap* and *Formation Energy*). We define semantic mappings based on MDO to interpret such data. All the semantic mappings are available at our repository.[2] We collect data in the sizes of 1K, 2K, 4K, 8K, 16K and 32K from each database to populate the five entities. The size 1K means 1000 entities of each entity type. We represent this data in different formats, such as tabular data for relational databases and for CSV files, and JSON-formatted data for JSON files. Additionally, for the RDF-based systems in our evaluation, we create an RDF file based on RML mappings and MDO for each dataset setting. We have six dataset settings for the experiments, which are 1K-1K, 2K-2K, 4K-4K, 8K-8K, 16K-16K and 32K-32K. Taking 32K-32K as an example, for each entity type, the test data contains the data in the size of 32K from the Materials Project and OQMD, respectively.

[2]`https://github.com/LiUSemWeb/OBG-gen/tree/main/mapping_parser/semantic_mappings`

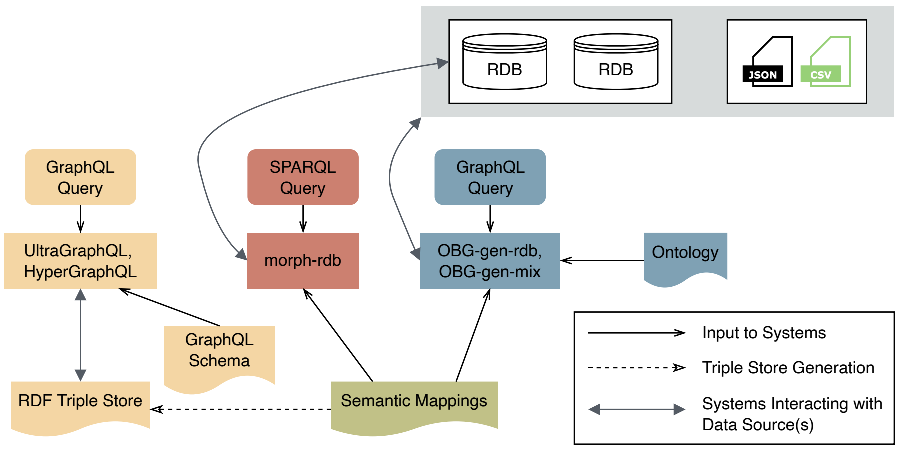


Figure 8.1: An outline of the evaluation.

8.1.2 Systems

In Figure 8.1, we show how the five systems are configured in the evaluation. HyperGraphQL and UltraGraphQL are provided with the same RDF data for each dataset setting. OBG-gen-rdb and morph-rdb are provided with two MySQL database instances hosting data from the Materials Project and OQMD respectively. Conceptually, OBG-gen-mix is also provided with two database instances. However, each instance contains different formats of data such as data in a MySQL database, or in CSV or JSON files. More detailed, the instance for Materials Project has *Composition* data in JSON format and *Band Gap* data in CSV format. The instance for OQMD has *Structure* and *Band Gap* data in JSON format and *Formation Energy* data in CSV format. The data representing other entities for each instance is stored in MySQL database instances.

8.1.3 Queries

We create queries that cover different features, aiming to evaluate our system based on qualitative aspects regarding what functionalities the system can satisfy and quantitative aspects regarding how the system performs over different data sizes. Query features of queries without and with filter expressions are shown in Table 8.1 and Table 8.2, respectively. All the queries correspond to complex competency questions stated in the requirements analysis of MDO as presented in Chapter 5. From the perspective of GraphQL, we consider which choke point a query covers. The details of choke points are introduced in LinGBM.[3] These choke points are regarding the key technical challenges. We characterize all queries using the perspectives of choke points, domain interest (*DI*), and result size (*RS*). *DI* indicates that the query is a domain-interest query. For *RS*, as the dataset grows, we consider whether the result size increases linearly (*L*) or more than linearly (*NL*), or stays a constant value (*C*). For queries with filter expressions we take into account the *filter expression form* and whether the filtering AST differs from the query AST (*Diffs*), such as in the example in Figure 4.4b where the filtering AST and the query AST are different.

Table 8.1: Features of queries without filter conditions.

Query	Choke Points	*Domain Interest (DI)*	*Result Size (RS)*
Q1	2.1, 2.2		*L*
Q2	2.1, 2.2	✓	*L*
Q3	1.1, 2.1, 2.2	✓	*L*
Q4	1.1, 2.1, 2.2	✓	*L*
Q5	2.2		*L*

Table 8.2: Features of queries with filter conditions.

Query	Choke Points	*DI*	*Diffs*	*filter expression form*	*RS*
Q6	1.1, 2.1, 2.2, 4.1, 4.4		✓	A	*C*
Q7	1.1, 2.1, 2.2, 4.1, 4.4		✓	A & B	*C*
Q8	1.1, 2.1, 2.2, 4.1, 4.4, 4.5	✓	✓	A & (B | C)	*C*
Q9	1.1, 2.1, 2.2, 4.1, 4.4, 4.5	✓	✓	A & B	*C*
Q10	1.1, 2.1, 2.2, 4.1, 4.4, 4.5	✓	✓	A & (B & C)	*NL*
Q11	2.2, 4.1, 4.4, 4.5		✓	(A & B) & ((A & B) | C)	*NL*
Q12	2.2, 4.1, 4.4	✓		A	*NL*

In Table 8.3, we show more details of meanings of different filter expressions for Q6–Q12. The filter expressions for Q6 and Q12 are more simple than those for Q7–Q11 where the filter expressions have sub-expressions connected by boolean operators. Query features in terms of *DI*, and the *filter expression form* can help us understand systems qualitatively; *Diffs* and *RS* help in understanding systems quantitatively in the scaling analysis over different

[3]`https://github.com/LiUGraphQL/LinGBM/wiki/Choke-Points`

data sizes. We show Q1 in Listing 8.1 and Q7 in Listing 8.3. The results of these two queries are given in Listing 8.2 and Listing 8.4, respectively. Q1 requests all the structures containing the reduced chemical formula of each structure composition. Q7 requests all the calculations where the ID is in a given list of values, and the reduced chemical formula is in a given list of values. All the 12 queries for our experiments are given in Appendix C.1.

Table 8.3: Meanings of filter expressions in Q6 to Q12.

Query	Filter expression meaning
Q6: A	id is in a list
Q7: A & B	id is in a list **and** reduced chemical formula is in a list
Q8: A & (B | C)	id is in a list **and** reduced chemical formula is in list a_1 **or** list a_2
Q9: A & B	property name is “Band Gap” **and** value is greater than 5
Q10: A & (B & C)	reduced chemical formula is in a list **and** property name is “Band Gap” **and** value is greater than 5
Q11: (A & B) & ((A & B) | C)	(property name is “Band Gap” **and** value is greater than 4) **and** ((property name is “Band Gap” **and** value is greater than 4) **or** reduced chemical formula is in a list)
Q12: A	reduced chemical formula contains silicon element

Listing 8.1: List all the structures containing the reduced chemical formula of each structure’s composition.

```
{
   StructureList{
       hasComposition{
           ReducedFormula
       }
   }
}
```

Listing 8.2: The JSON response (an excerpt) of the query in Listing 8.1.

```
{
  "data": {
    "StructureList": [
      { "hasComposition": { "ReducedFormula": "CeCrS2O" } },
      { "hasComposition": { "ReducedFormula": "TlP(HO2)2" } },
      { "hasComposition": { "ReducedFormula": "YClO" } }
    ]
  }
}
```

8

Listing 8.3: List all the calculations where the ID is in a given list of values and the reduced chemical formula is in a given list of values.

```
{
  CalculationList(
    filter: {
      _and: [
        {
          ID: {
            _in: [ "6332","8088","21331","mp-561628","mp-614918" ]
          }
        }
        {
          hasOutputStructure: {
            hasComposition: {
              ReducedFormula: {
                _in: [ "MnCl2","YClO" ]
              }
            }
          }
        }
      ]
    }
  )
  {
    ID
    hasOutputCalculatedProperty {
      PropertyName
      numericalValue
    }
  }
}
```

Listing 8.4: The JSON response of the query in Listing 8.3.

```
{
  "data": {
    "CalculationList": [
      {
        "ID": "6332",
        "hasOutputCalculatedProperty": [
          {
            "PropertyName": "Formation Energy",
            "numericalValue": -1.3247
          },
          {
            "PropertyName": "Band Gap",
            "numericalValue": 1.807
          }
        ]
      },
      {
        "ID": "mp-614918",
        "hasOutputCalculatedProperty": [
          {
            "PropertyName": "Formation Energy",
            "numericalValue": -40.6691
          },
          {
            "PropertyName": "Band Gap",
            "numericalValue": 2.2287
          }
        ]
      }
    ]
  }
}
```

8.1.4 Experiments and measurements

We evaluate the query execution time (QET) of the different systems over the six dataset settings. Separately for each query, we run the query four times and always consider the first run to be a warm-up, then take the averaged value of the remaining three runs. Figure 8.2 and Figure 8.3 illustrate the measurements for all systems and queries per data size. Figure 8.4 to Figure 8.15 illustrate the measurements over the six data sizes per query (Q1–Q12). The measures for all data sizes and all queries are available online.[4] For UltraGraphQL, we have measurements only for queries Q1–Q4 because UltraGraphQL does not support queries with filtering conditions. For HyperGraphQL answering queries with filter expressions, we have only the measurement for Q6 because the system can only deal with filtering by resource IRIs. Additionally, Table 8.4 illustrates a comparison between OBG-gen-rdb and morph-rdb.

[4] `https://github.com/LiUSemWeb/OBG-gen/tree/main/evaluation`

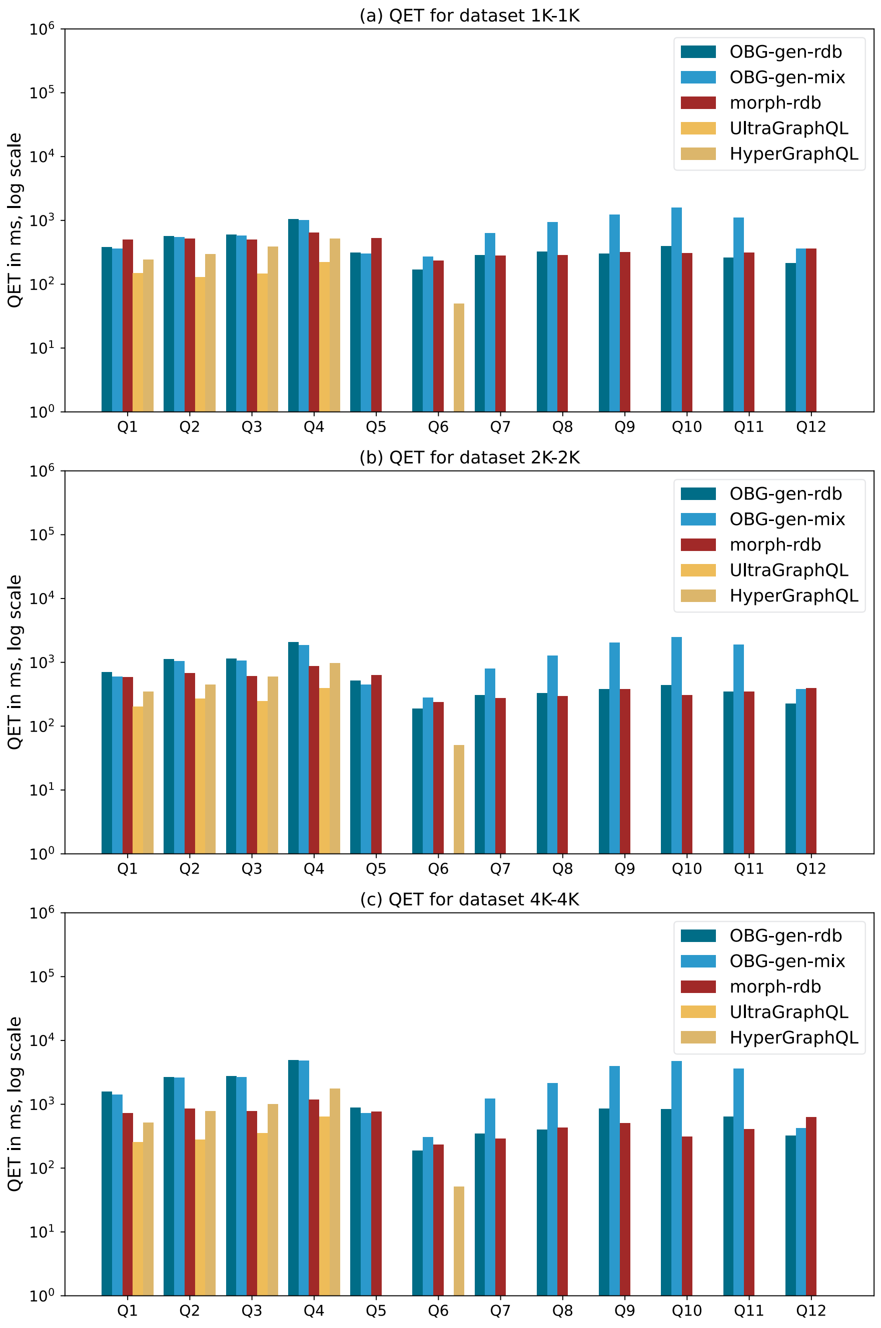


Figure 8.2: Query Execution Time (QET) for data size (1K-1K, 2K-2K, 4K-4K) on materials datasets.

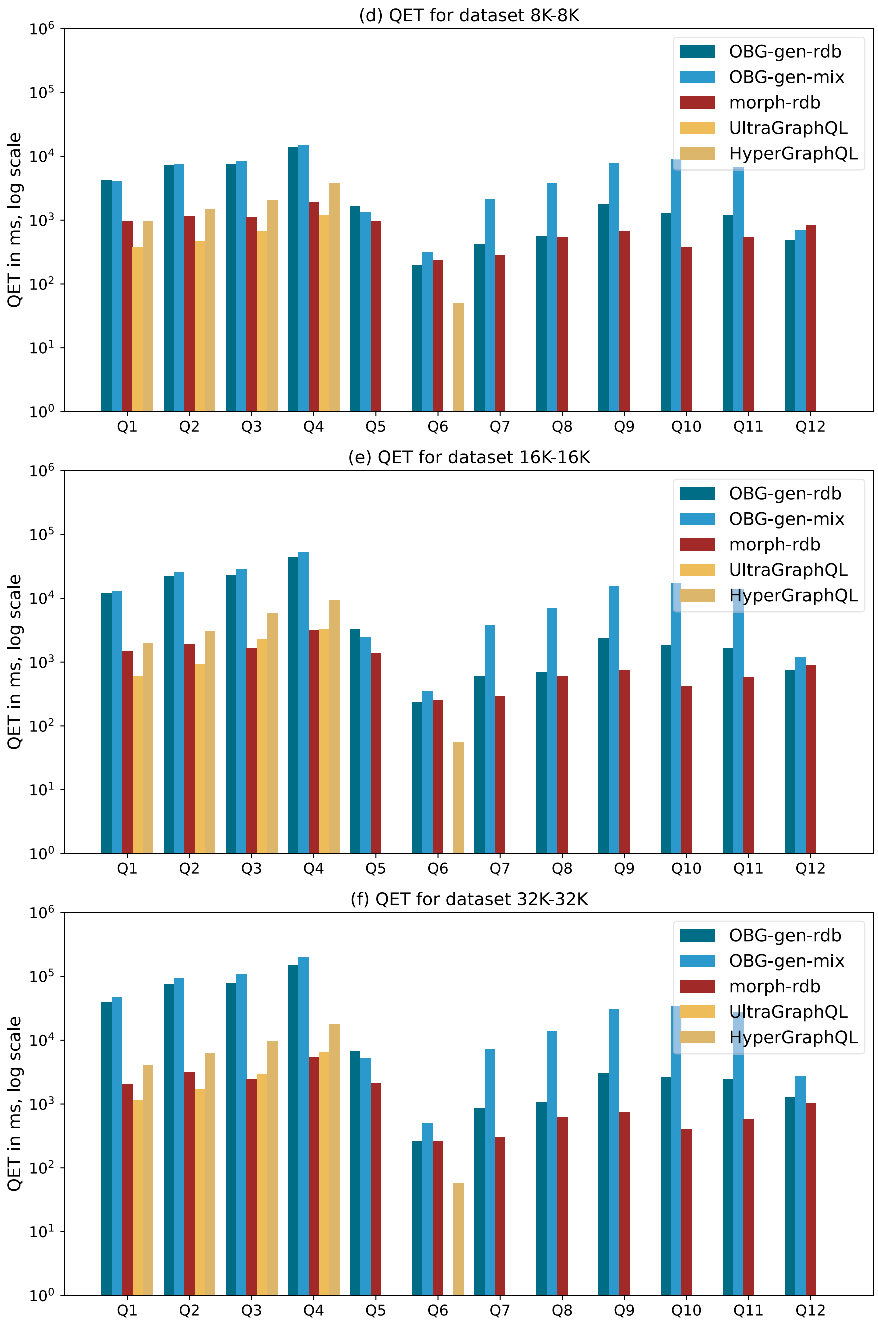


Figure 8.3: Query Execution Time (QET) for data size (8K-8K, 16K-16K, 32K-32K) on materials datasets.

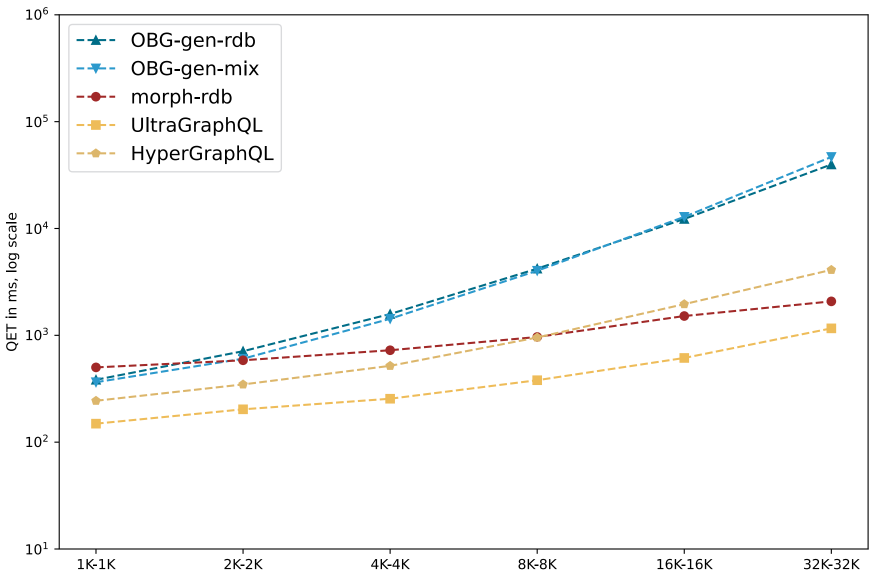


Figure 8.4: Query Execution Time (QET) for Q1 on materials datasets.

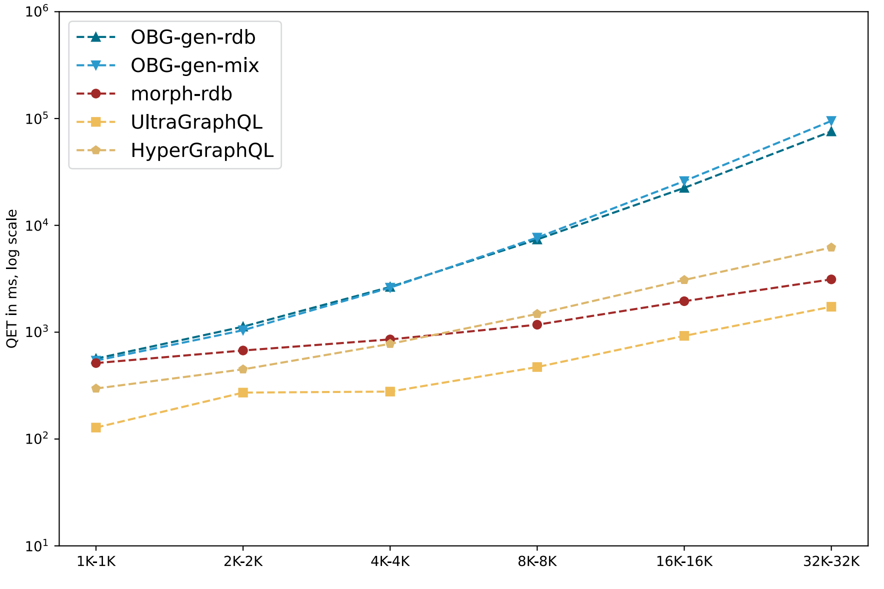


Figure 8.5: Query Execution Time (QET) for Q2 on materials datasets.

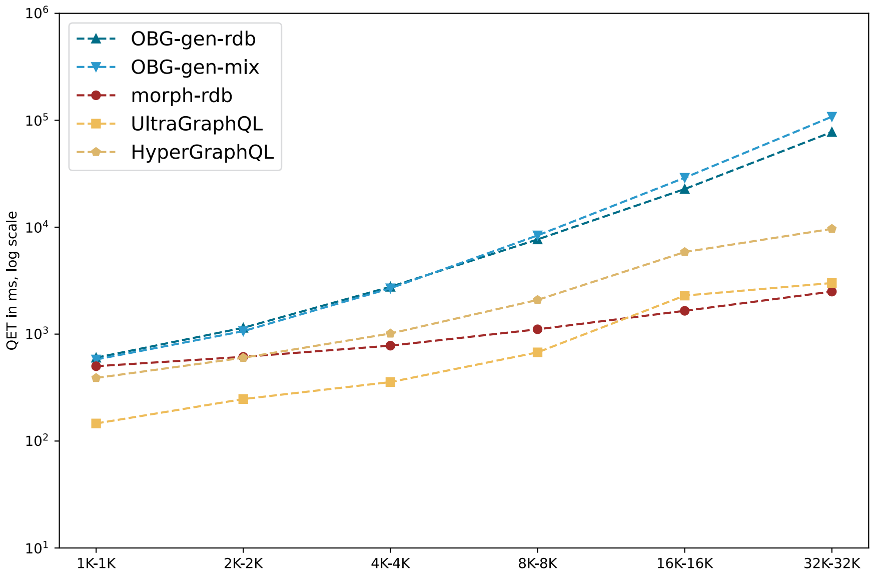


Figure 8.6: Query Execution Time (QET) for Q3 on materials datasets.

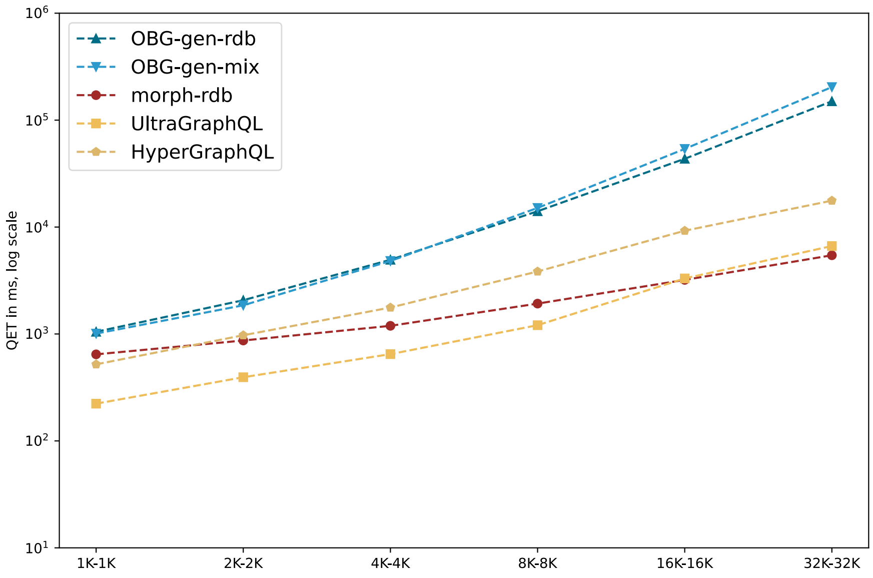


Figure 8.7: Query Execution Time (QET) for Q4 on materials datasets.

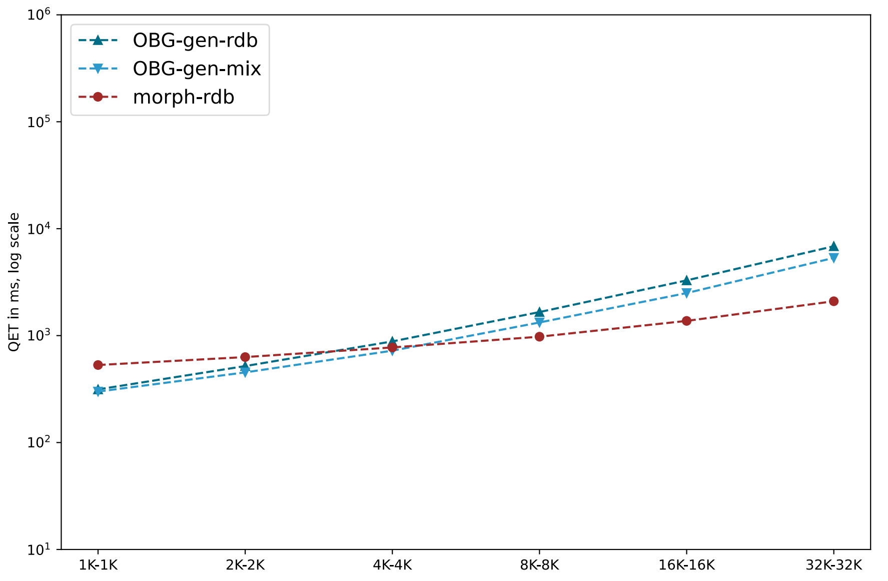


Figure 8.8: Query Execution Time (QET) for Q5 on materials datasets.

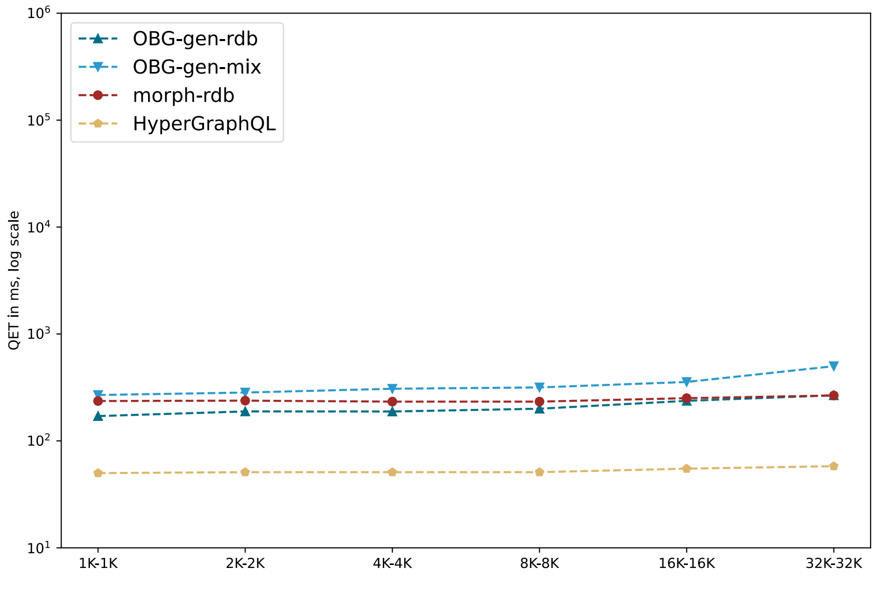


Figure 8.9: Query Execution Time (QET) for Q6 on materials datasets.

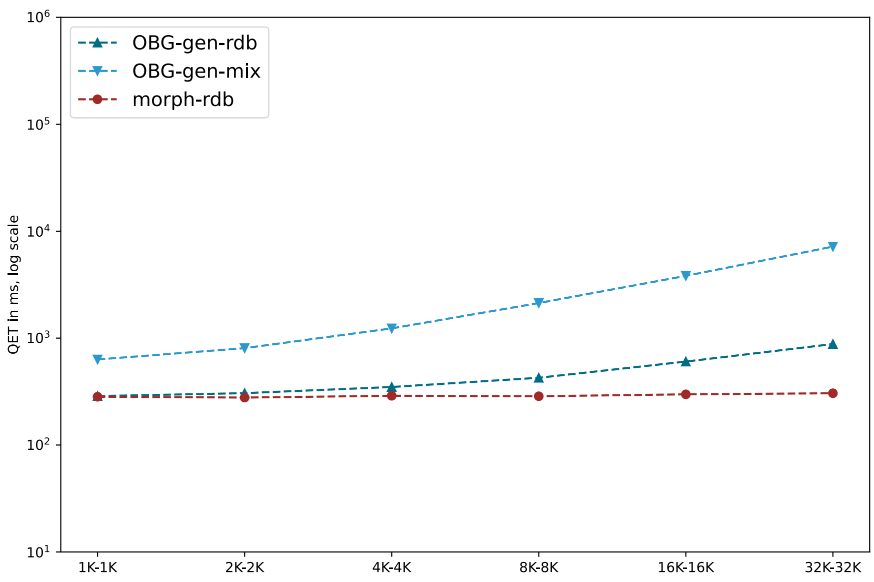


Figure 8.10: Query Execution Time (QET) for Q7 on materials datasets.

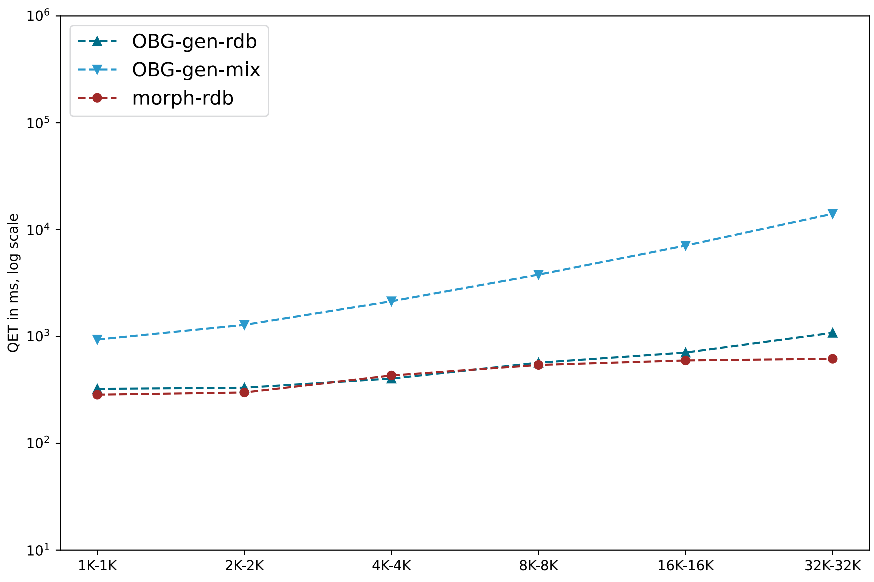


Figure 8.11: Query Execution Time (QET) for Q8 on materials datasets.

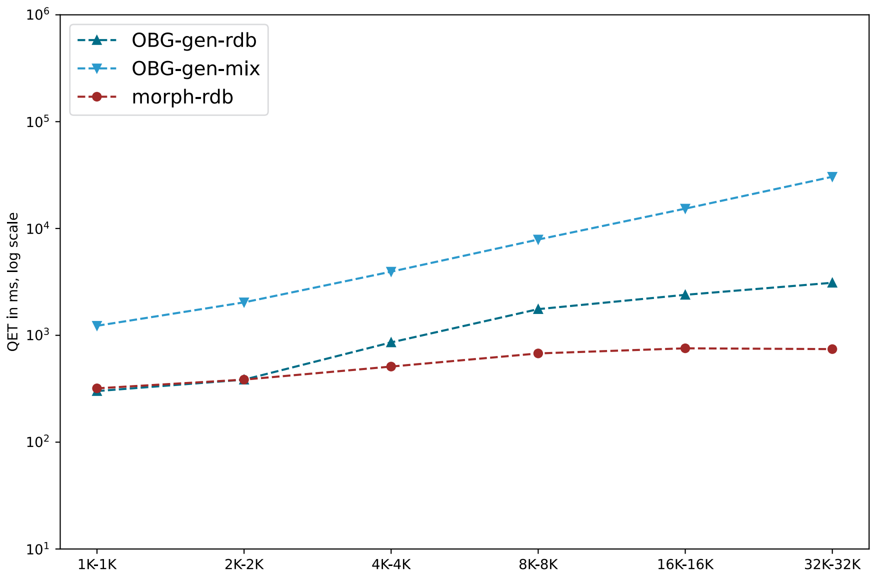


Figure 8.12: Query Execution Time (QET) for Q9 on materials datasets.

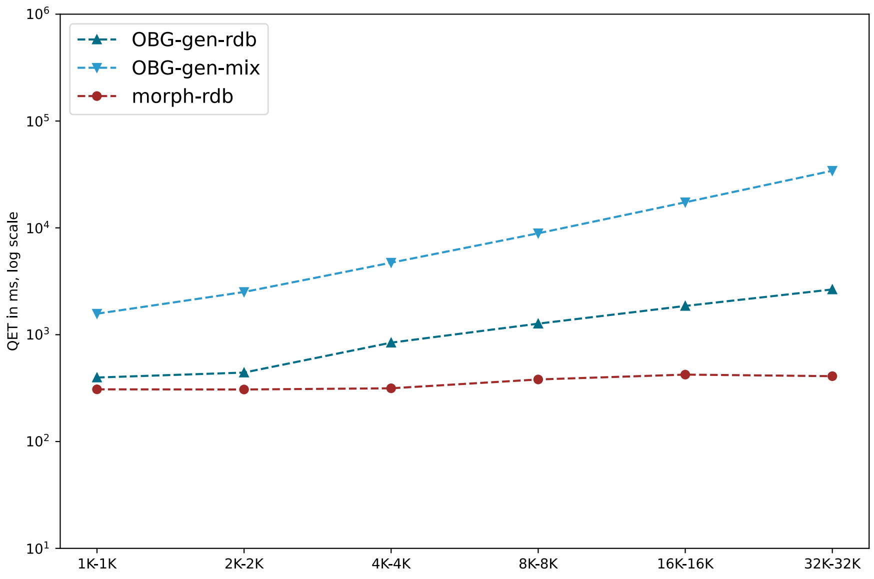


Figure 8.13: Query Execution Time (QET) for Q10 on materials datasets.

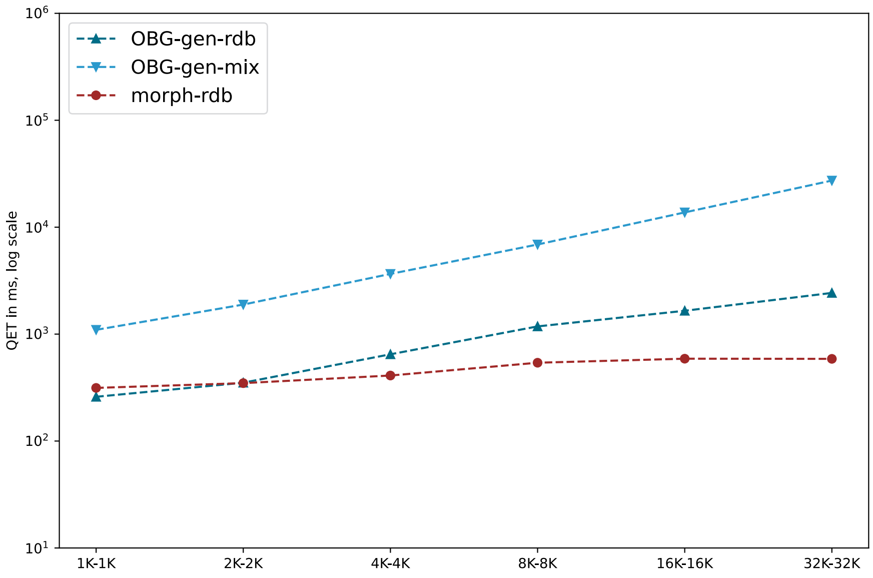


Figure 8.14: Query Execution Time (QET) for Q11 on materials datasets.

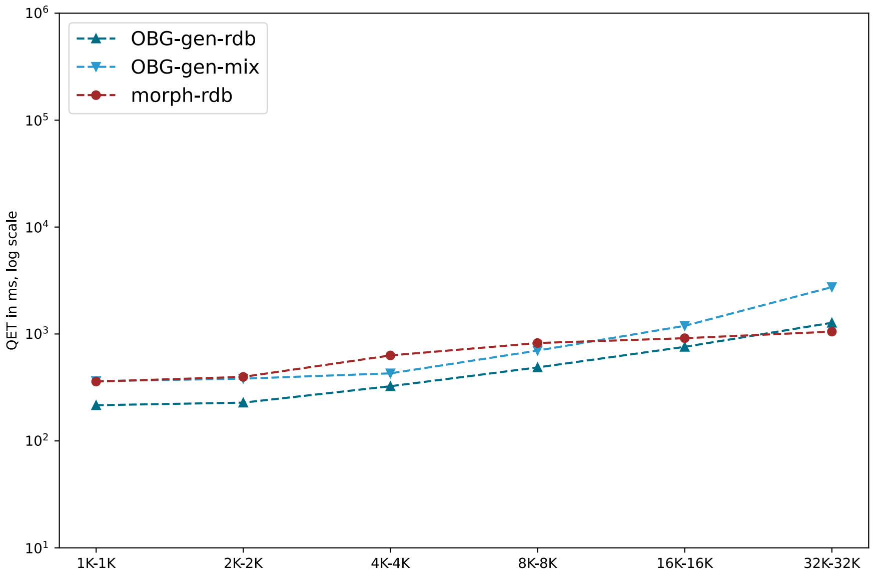


Figure 8.15: Query Execution Time (QET) for Q12 on materials datasets.

8.1.5 Results and discussion

By analyzing the obtained measurements, we summarize three observations.

The **first** observation is that both GraphQL servers generated by OBG-gen-rdb and OBG-gen-mix can answer all 12 of the queries covering different features (such as choke points) and corresponding to competency questions of MDO. Therefore, the framework presented in Chapter 3 is feasible for data access and integration; this answers **RQa** and **RQb**. Particularly, the GraphQL schema generated based on the ontology can provide an (integrated) view of underlying (heterogeneous) data; the generic resolver function based on the semantic mappings is capable of accessing heterogeneous data sources, combining the retrieved data (which may be in different formats), and structuring the data according to the GraphQL schema.

The **second** observation is regarding queries without filtering conditions (Q1–Q5) (cf. Figure 8.4 to Figure 8.8). All of the systems have increases of QETs as the size of the dataset increases. However, morph-rdb is less sensitive to the data size increase compared with other systems. UltraGraphQL and HyperGraphQL outperform other systems for some smaller datasets (e.g., HyperGraphQL's QETs of Q1 and Q2 for datasets, UltraGraphQL's QETs for Q1 from 1K-1K to 4K-4K). We explain this by the fact that these two systems have additional context information declaring URIs of classes to which instances in the RDF data belong (as shown in Table 4.1 in Chapter 4), which is unlike the other systems which have to make use of semantic mappings to output queries to be evaluated against the underlying data sources. OBG-gen-rdb can outperform morph-rdb for some queries in smaller datasets (e.g., Q1 in 1K-1K, Q5 in 1K-1K and 2K-2K as shown in Table 8.4). For some queries, OBG-gen-rdb and morph-rdb have close QETs (e.g., Q2 in 1K-1K as shown in Table 8.4).

The **third** observation is regarding how OBG-gen-rdb and morph-rdb perform for queries with filter conditions (Q6–Q12) (cf. Figure 8.9 to Figure 8.15). The two systems behave similarly for Q6 with stable QETs and Q12 with slight increases, as the data size increases. As Table 8.2 shows, the result size of Q6 shown in Appendix C.6 is a constant over all the datasets in different sizes. Additionally, as shown in Table 8.3 the filter expressions for Q6 and Q12 are simpler compared with those of Q7–Q11. Therefore, the QETs consumed for evaluating filtering expressions for Q6 and Q12 are less than those of Q7–Q11. For other queries (Q7–Q11), morph-rdb outperforms OBG-gen-rdb, however

Table 8.4: Comparison between OBG-gen-rdb and morph-rdb (QET in seconds).

Data	System	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
1K-1K	OBG-gen-rdb	**0.3**	0.5	0.6	1.0	**0.3**	**0.1**	0.2	0.3	0.3	0.3	**0.2**	**0.2**
	morph-rdb	0.5	0.5	**0.5**	**0.6**	0.5	0.2	0.2	**0.2**	0.3	**0.5**	0.3	0.3
2K-2K	OBG-gen-rdb	0.7	1.1	1.1	2.0	**0.5**	**0.1**	0.3	0.3	0.3	0.4	0.3	**0.2**
	morph-rdb	**0.5**	**0.6**	**0.6**	**0.8**	0.6	0.2	**0.2**	**0.2**	0.3	**0.3**	0.3	0.3
4K-4K	OBG-gen-rdb	1.5	2.6	2.7	4.9	0.8	**0.1**	0.3	0.4	0.8	0.8	0.6	**0.3**
	morph-rdb	**0.7**	**0.8**	**0.7**	**1.1**	**0.7**	0.2	**0.2**	0.4	**0.5**	**0.3**	**0.4**	0.6
8K-8K	OBG-gen-rdb	4.2	7.3	7.6	14.0	1.6	0.2	0.4	0.5	1.7	1.2	1.1	**0.4**
	morph-rdb	**0.9**	**1.1**	**1.1**	**1.9**	**0.9**	0.2	**0.2**	0.5	**0.6**	**0.3**	**0.5**	0.8
16K-16K	OBG-gen-rdb	12.2	22.4	22.7	43.5	3.2	0.2	0.6	0.7	2.3	1.8	1.6	**0.7**
	morph-rdb	**1.5**	**1.9**	**1.6**	**3.2**	**1.3**	0.2	**0.2**	**0.5**	**0.7**	**0.4**	**0.5**	0.9
32K-32K	OBG-gen-rdb	39.7	75.7	77.5	149.9	6.8	0.2	0.8	1.0	3.1	2.6	2.4	1.2
	morph-rdb	**2.0**	**3.1**	**2.4**	**5.4**	**2.0**	0.2	**0.3**	**0.6**	**0.7**	**0.4**	**0.5**	**1.0**

the differences between the two systems are less than those for queries without filtering conditions (e.g., Q1–Q4). The filtering conditions in GraphQL queries for OBG-gen-rdb and in SPARQL queries for morph-rdb are written within *WHERE* clauses in SQL queries, thus will be evaluated against the back-end databases. The similar observation is also found in [69] where the experiment metrics shows that morph-rdb outperforms other systems (e.g., morph-morphql) as the size of dataset increase due to the SPARQL to SQL optimizations [25].

Based on the second and the third observations, we can answer the research question **RQc**. The GraphQL servers generated by OBG-gen performs similarly compared with other systems for queries without filtering conditions, but are more sensitive to the increase of datasets even they can outperform for some queries in smaller datasets. By comparing OBG-gen-rdb and morph-rdb, we summarize the reasons as follows. As shown in Chapter 4, the implementation of OBG-gen is based on representing a GraphQL query with abstract syntax trees (e.g., Figure 4.4 in Chapter 4) and processing a referencing object map from semantic mappings in a nested loop (e.g., line 22 to line 29 in Algorithm 3). In this way, two basic requests are sent to underlying data sources to get the data with respect to *parent triples map* and *current triples map* as shown in Section 4.2.2 of Chapter 4, and there is a join operation locally in our implementation (e.g., line 29 in Algorithm 3). For instance, to answer Q7 shown in Figure 8.3, as the query asks for a list of `Calculations` and for each `Calculation` asks for the `ID` field of which the returned type is scalar and the `hasOutputCalculatedProperty` field of which the returned type is a list of `CalcualtedProperty`, therefore two requests are sent to underlying data sources to get the data for populating `ID`, and `PropertyName` and `numericalValue`, respectively. While for morph-rdb, based on semantic mappings, a SPARQL query is translated to a single SQL query. For queries with filtering conditions, both OBG-gen-rdb and morph-rdb can take the advantages of rewriting filter conditions into SQL queries so that the increases of QETs as data size increases are not obvious.

8.2 Evaluation based on LinGBM

To show the generalizability of our system, we conduct an evaluation based on LinGBM. It is developed as a performance benchmark for GraphQL server implementations. LinGBM provides tools for generating datasets (data generator)[5] and queries (query generator),[6] and for testing execution time and response time (test driver).[7]

8.2.1 Data

The dataset generated by the data generator is a scalable, synthetic dataset regarding the *University* domain, including several entity types (e.g., universities and departments). We generate data in scale factors (*sf*) 4, 20 and 100. We then create three MySQL database instances to store the data in these three scale factors, respectively. We use a modified version of the GraphQL schema provided by LinGBM for our GraphQL server, and define RML mappings according to the work in morph-graphql[8] [69]. The modification part is regarding input object type definitions so that we can use input objects to represent filtering conditions as we show in Chapter 3 and Chapter 4. The entire GraphQL schema is shown in Appendix B.2.

8.2.2 Queries

The experiments are performed over eight query sets, where each set contains 100 queries that are generated using the LinGBM query generator based on

[5]https://github.com/LiUGraphQL/LinGBM/tree/master/tools/datasetgen
[6]https://github.com/LiUGraphQL/LinGBM/tree/master/tools/querygen
[7]https://github.com/LiUGraphQL/LinGBM/tree/master/tools/testdriver_QET_QRT
[8]https://github.com/oeg-upm/morph-graphql/tree/master/examples/LinGBM-v2

a query template (QT). A query template has placeholders where each placeholder represents that an input argument can be assigned. The query generator can generate a set of actual queries (query instances) based on a query template in which the placeholder in the query template is replaced by an actual value. We select eight query templates (QT1–QT6, QT10 and QT11) for constructing these eight query sets (QS1–QS8). We show an example query according to QT5 in Listing 8.5. For each query set, we show an example query in Appendix C.2. The other six query templates from LinGBM requires GraphQL servers to have implementations for functionalities such as ordering and paging which are not considered currently by OBG-gen. However, these functionalities are interesting for future extension of OBG-gen.

8.2.3 Experiments, results and discussion

Same as the real case evaluation, we evaluate the query execution time (QET) of our system on the three datasets. Each query from a query set is evaluated once. We show the average query execution times for the different query sets in Table 8.5. Based on the obtained measurements, we observe that our system has slight increases for QS1, QS2, QS4, QS6 and QS7 in terms of the average QETs. For QS3, the average QET is stable for all the three datasets. For QT5, the increase from 0.51 seconds at data scale factor 20 to 13.85 seconds at data scale factor 100 is due to the dramatic increase in result size. More specifically, the queries in QS5 and QS8 need to access the *'graduateStudent'* table which increases dramatically in size from 50,482 (sf=20) to 252,562 (sf=100). This is the reason for the average QET of QS8 increasing in sf=100. Additionally, each query in QS5 repeats a cycle two times (*'university'* to *'graduateStudent'* to *'university'*) and requests the students' emails and addresses along the way. This causes the larger increase in average QET of QS5. The above synthetic experiments indicate that our system can work in a general domain.

Table 8.5: Average QET (in seconds).

sf	QS1 (QT1)	QS2 (QT2)	QS3 (QT3)	QS4 (QT4)	QS5 (QT5)	QS6 (QT6)	QS7 (QT10)	QS8 (QT11)
4	0.11	0.13	0.12	0.15	0.19	0.13	0.10	0.26
20	0.12	0.15	0.12	0.18	0.51	0.15	0.18	0.90
100	0.15	0.27	0.12	0.26	13.85	0.23	0.72	4.41

Listing 8.5: A query according to query template 5.

```
{
  DepartmentList(
    filter:{
      nr: { _eq: 314 }
    })
  {
    nr
    subOrganizationOf {
      nr
      undergraduateDegreeObtainedBystudent {
        nr
        emailAddress
        memberOf {
          nr
          subOrganizationOf {
            nr
            undergraduateDegreeObtainedBystudent {
              nr
              emailAddress
              memberOf {
                nr
              }
            }
          }
        }
      }
    }
  }
}
```

8.3 Summary

In this chapter, we have conducted an evaluation of the GraphQL-based framework for data access and integration presented in Chapter 3. We use our prototype, OBG-gen, as presented in Chapter 4, to generate GraphQL servers. We conduct a real case evaluation over data collected from two databases in the materials design domain. In addition, we evaluate our approach based on a synthetic dataset. In the next chapter, we show the application of our approach for the community effort, *Open Databases Integration for Materials Design* (OPTIMADE).

Chapter 9

An application to OPTIMADE

As previously mentioned, the OPTIMADE (*Open Database Integrations for Materials Design*) API specification is one of the inspirations upon which this thesis has been constructed. The collaborative effort of materials databases in OPTIMADE is to develop a specification for a common REST API. Such a common API specifies how data can be retrieved. In this regard, each database provider within the OPTIMADE consortium provides a way for users to access its data in accordance with this common API.

In Chapter 5, we have shown the vision of the usage of Materials Design Ontology (MDO). One common usage is for data integration and access through MDO-based mediation. In Chapter 3, we have outlined a GraphQL-based framework for data access and integration with a prototype implementation in Chapter 4. Furthermore, in Chapter 8, we have shown experiments in the materials design domain, in which we make use of MDO to define semantic mappings for datasets collected from the Materials Project and OQMD, and to set up a GraphQL server using OBG-gen (Ontology-Based GraphQL Server Generation). To apply our approach to OPTIMADE, we focus on (i) how the data following the OPTIMADE API can be annotated using MDO terminology, (ii) comparing the GraphQL API, in which the GraphQL server is generated by OBG-gen using MDO, to the OPTIMADE API. As the OPTIMADE API is under development, our application is at the level of a proof of concept. In Section 9.1, we introduce the OPTIMADE API specification.

Then in Section 9.2 we introduce the usage of MDO and OBG-gen to OPTIMADE.

9.1 The OPTIMADE API

The OPTIMADE API provides a standard for how underlying materials databases can share data in a common manner. The consensus among database providers with the OPTIMADE consortium is that each database provider should have an endpoint, so that users can access the data through the OPTIMADE API. For instance, the Materials Project has the base URL, `https://optimade.materialsproject.org` and the OQMD has the base URL, `http://oqmd.org/optimade`.

The latest stable version of this API is v1.1.0.[1] Furthermore, a python library named *optimade-python-tools* has been developed in order for different data providers to share their data in accordance with the data model following the OPTIMADE API specification [159]. The OPTIMADE API specification defines a number of entries that users can use for accessing data. In Table 9.1, we list these entries and related properties.

Table 9.1: The entries and properties in OPTIMADE API specification.

Entry	Fields
Structure	id, type, immutable_id, elements, nelements, elements_ratios, chemical_formula_descriptive, chemical_formula_reduced, chemical_formula_hill, chemical_formula_anonymous, dimension_types, nperiodic_dimensions, lattice_vectors, cartesian_site_positions,nsites, species_at_sites, species, assemblies, structure_features
Reference	id, type, immutable_id, authors, year, title, journal, doi, etc.
Calculation	id, type, immutable_id, etc.

In Listing 9.1, we show an excerpt of the JSON response from a request that conforms to the OPTIMADE API. The endpoint in this case is provided by the Materials Project. The request url is `http://optimade.materialsproject.org/v1/structures?page_limit=100&filter=chemical_formula_reduced="MgNi"`, which retrieves structures in which the reduced chemical formula is MgNi.

[1]`https://petstore.swagger.io/?url=https://raw.githubusercontent.com/Materials-Consortia/OPTIMADE/master/schemas/openapi_schema.json`

Listing 9.1: An excerpt of the JSON response based on OPTIMADE API.

```json
{
  "data": [
    {
      "id": "mp-1010953",
      "type": "structures",
      "attributes": {
        "elements": ["Mg", "Ni"],
        "nelements": 2,
        "elements_ratio": [0.5, 0.5],
        "chemical_formula_descriptive": "MgNi",
        "chemical_formula_reduced": "MgNi",
        "chemical_formula_hill": "MgNi",
        "chemical_formula_anonymous": "AB",
        "dimension_types": [1, 1, 1],
        "nperiodic_dimensions": 3,
        "lattice_vectors": [
          [3.046453, 0.0, 0.0],
          [0.0, 3.046453, 0.0],
          [0.0, 0.0, 3.046453]
        ],
        "cartesian_site_positions": [
          [0.0, 0.0, 0.0],
          [1.5232265, 1.5232265, 1.5232265]
        ],
        "nsites": 2,
        "species": [
         {
          "name": "Mg",
          "chemical_symbols": ["Mg"],
          "concentration": [1]
         },
         {
          "name": "Ni",
          "chemical_symbols": ["Ni"],
          "concentration": [1]
         }
       ],
       "species_at_sites": ["Mg", "Ni"]
      }
    }
  ]
}
```

9.2 The usage of MDO and OBG-gen with OPTIMADE

Figure 9.1 illustrates the application of MDO to annotate the structure illustrated in Listing 9.1. As a convenience for readers, we show only one instance for a concept that has multiple instances within the instantiation. For all the keys labeled in blue in Listing 9.1, we can interpret their corresponding values using the MDO terminology. For those keys marked in yellow, *nelements*, *dimension_types*, *nperiodic_dimensions* and *nsites*, their values cannot be interpreted using the terminology in MDO of the current version 1.0. MDO can, however, interpret them if it models several data properties that are associated with the *Structure* class in the ontology. This will be taken into consideration in the future development of MDO.

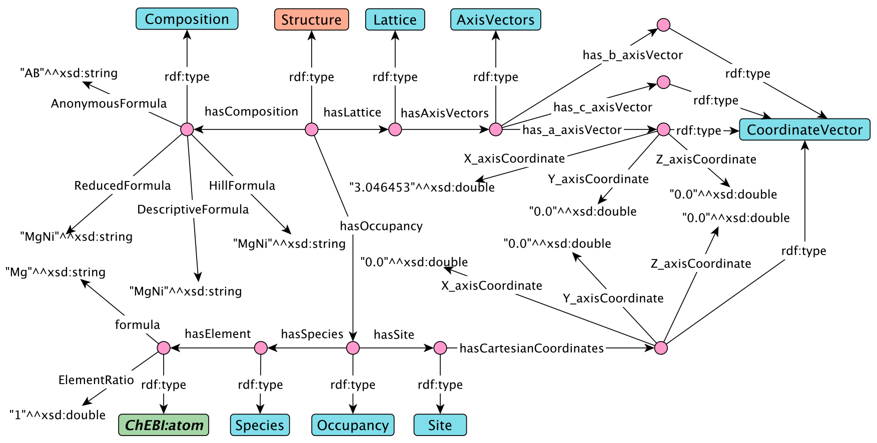


Figure 9.1: An instantiation of the structure shown in Listing 9.1.

In addition, we define semantic mappings using RML for annotating responses from OPTIMADE API requests using the MDO terminology. Based on these semantic mappings and the GraphQL schema shown in Appendix B.1, we use OBG-gen to generate a GraphQL server that can answer GraphQL queries in which the underlying data follows the OPTIMADE API specification.[2] We show a query example in Listing 9.2 and the corresponding result in Listing 9.3. This query also retrieves structures in which the reduced chemical formula is MgNi, just as the request does to get the data as shown

[2]The code for translating a OBG-gen supported filter conditions to OPTIMADE supported filter conditions is available at `https://github.com/LiUSemWeb/OBG-gen/tree/optimade-impl`.

in Listing 9.1. The key difference is that the GraphQL API allows users to specify particular fields that they want returned. For instance, in Listing 9.2 the query asks for two composition-related fields (*ReducedFormula* and *DescriptiveFormula*) but only one of the three vectors that represent a lattice (*has_a_axisVector*), in particular. The GraphQL API is therefore more flexible from a user's perspective. Another example, asking for structures of which the anonymous chemical formulas are "AB", is shown in Listing 9.4. Instead of asking for both composition-related fields and lattice-related fields like the query in Listing 9.2, this query just asks for composition-related fields. The query result is shown in Listing 9.5.

Listing 9.2: An example query over data following OPTIMADE API specification retrieving both composition related and lattice related fields.

```
{
  StructureList(
    filter:{
      hasComposition:{
        ReducedFormula:{
          _eq: "MgNi"
        }
      }
    }
  ){
    hasComposition{
      ReducedFormula
      DescriptiveFormula
    }
    hasLattice{
      hasAxisVectors{
        has_a_axisVector{
          X_axisCoordinate
          Y_axisCoordinate
          Z_axisCoordinate
        }
      }
    }
  }
}
```

Listing 9.3: The result of the query in Listing 9.2.

```
{
  "data": {
    "StructureList": [
      {
        "hasComposition": {
          "DescriptiveFormula": "MgNi",
          "ReducedFormula": "MgNi"
        },
        "hasLattice": {
          "hasAxisVectors": {
            "has_a_axisVector": {
              "X_axisCoordinate": 3.046453,
              "Y_axisCoordinate": 0,
              "Z_axisCoordinate": 0
            }
          }
        }
      }
    ]
  }
}
```

Listing 9.4: An example query over data following OPTIMADE API specification retrieving composition related fields.

```
{
  "StructureList"(
    filter:{
      hasComposition:{
        AnonymousFormula:{
          _eq:"AB"
        }
      }
    }
  ){
    hasComposition{
      ReducedFormula
      DescriptiveFormula
    }
  }
}
```

Listing 9.5: The result of the query in Listing 9.4.

```
{
  "data": {
    "StructureList": [
      {
        "hasComposition": {
          "DescriptiveFormula": "AuN",
          "ReducedFormula": "AuN"
        }
      },
      {
        "hasComposition": {
          "DescriptiveFormula": "MgNi",
          "ReducedFormula": "MgNi"
        }
      },
      {
        "hasComposition": {
          "DescriptiveFormula": "HTi",
          "ReducedFormula": "HTi"
        }
      },
      {
        "hasComposition": {
          "DescriptiveFormula": "Mo2N2",
          "ReducedFormula": "MoN"
        }
      },
      {
        "hasComposition": {
          "DescriptiveFormula": "OPd",
          "ReducedFormula": "OPd"
        }
      },
      {
        "hasComposition": {
          "DescriptiveFormula": "Mg3Sn3",
          "ReducedFormula": "MgSn"
        }
      },
      {
        "hasComposition": {
          "DescriptiveFormula": "Au4Pr4",
          "ReducedFormula": "AuPr"
        }
        },
        {
          "hasComposition": {
            "DescriptiveFormula": "MnZn",
            "ReducedFormula": "MnZn"
          }
        }
      ]
    }
}
```

9.3 Summary

In this chapter, we have introduced an application to OPTIMADE in terms of the usage of the GraphQL-based framework and MDO. Due to the fact that the OPTIMADE API is under development, our application is at the level of a proof of concept.

CHAPTER

10 Limitations and future work

In the previous chapters, we presented a GraphQL-based framework for data access and integration, introduced different efforts aiming at enabling GraphQL server generation within the framework, and showed the evaluations and applications. There are still some limitations, which can be resolved in the future. Additionally, based on our experience working in the interdisciplinary space between the Semantic Web field and the materials design field, we show additional directions for future research.

10.1 Towards more user-friendly data access, data integration and ontology extension

In Chapter 3, we have presented a GraphQL-based framework for data access and integration, which includes the GraphQL server generation process and the GraphQL query answering process. Ontologies and semantic mappings are essential to enable the automatic generation of GraphQL servers. Therefore, the coverage and the scope of the ontology and semantic mappings are important and their definition depends on the users or developers who are involved in the GraphQL server generation process. This means that when new data sources are added to databases, or new types of data are added, new semantic mappings need to be defined. It may also be necessary to modify the ontology if we need to add additional concepts or relationships covering semantics that can be used to annotate the added data. However, there is not much work on providing users and developers with suitable tools for maintaining semantic mappings in a data integration scenario (as discussed in [47]). The same issue exists when both ontologies and semantic mappings are required in a data integration scenario. Thus, it would be interesting to investigate this problem and to investigate what the functionalities that are required in such a tool in the future research.

In addition, our current effort of ontology-based GraphQL schema generation focuses on GraphQL language features that support semantics-aware and integrated data access, namely how underlying data can be queried, rather than reflecting the semantics of a complex knowledge representation language in the context of GraphQL schemas. Therefore, not all description logic constructors are used, but rather only those that are necessary for data access via GraphQL. It would be worthwhile to investigate how to represent more complex description logic constructors within the GraphQL context.

In Chapter 6, we presented an approach for extension of domain ontologies and conducted experiments with a domain expert and two ontology engineers regarding extension of domain ontologies. However, for the application of this approach in practice for specific domains, a user interface would be necessary to allow domain experts to use the approach effectively. We have implemented a prototype based on ToPMine-FTCA in [160], which currently provides a user interface for users to validate phrases and extend an ontology. Directions for future work include conducting experiments in more domains based on this prototype, and updating ToPMine-FTCA if needed.

10.2 Limitations in mapping languages

In our work, we use RML because it has the ability to support more data formats (e.g., data in relational databases, JSON-formatted or CSV-formatted data). In addition to this, other mapping languages are designed to deal with specific data formats (e.g., R2RML is suitable for data in relational databases). Despite the flexibility provided by RML when it comes to data formats, it is limited in some cases. For instance, as we describe in Section 4.2.2 of Chapter 4, a referencing object map refers to another triples map (called a parent triples map) by using a `rr:joinCondition` property to state the join condition between the current triples map and the parent triples map, in which the join condition contains two properties `rr:child` and `rr:parent` of which the values must be logical references to logical sources of the current triples map and the parent triples map, respectively. Therefore, when we need to define a referencing object map using RML to interpret the underlying data, the underlying data must contain references (columns in relational data or CSV-formatted data, key fields in JSON-formatted data) whose values can be used for joining. Otherwise, even if we are able to annotate the underlying data with terminologies from ontologies, we would not be able to use RML mappings to materialize the data or use a virtual-based approach to access or integrate the data. Similarly, this problem exists in other mapping languages, such as R2RML. Additionally, current mapping languages lack formalization and are associated with specific engines [47]. As a result, such mapping languages are difficult to extend and it is difficult to make them interoperable.

10.3 Semantic Web meets Materials Science

Although this thesis presents a framework of ontology-driven data access and integration with an application in the materials design domain, there are still a number of challenges that exist when employing Semantic Web-based technologies in the materials science domain. One group of challenges relates to the representation of domain knowledge in materials science. Currently, the Materials Design Ontology effort focuses on computational methods and structures at basic microscopic time and space scales. However, designing a material with a set of expected properties involves the design not only on the microscopic scale, but also on the macroscopic scale. When materials design processes at all levels must be integrated and automated, which is the goal of the materials science domain, we need to consider how ontologies representing different levels of domain knowledge can work together without conflicts. A direction for future work is to research on how to represent the fundamental domain knowledge that can fit into different levels of materials science and engineering.

In addition, many research groups in the field are developing ontologies that target different sub-domains, such as materials design and materials experiments. These domains are not orthogonal and may share some general concepts and relations. Unlike the biomedical domain, which has had quite a lot of domain ontologies created over the decades and gains experiences in ontology alignment (e.g. the work in [161] summarized experiences from aligning two representative ontologies in the biomedical domain), there is not much work focusing on ontology alignment in the materials science field. However we can foresee that the need for aligning ontologies in the materials science domain will arise. It is a challenge that there is no formally defined knowledge base or thesaurus that can be used for ontology alignment systems. Therefore, we should develop methods for building background knowledge bases or thesauri automatically through the learning of ontologies, or semi-automatically through the contribution of domain experts. The Ontology Alignment Evaluation Initiative (OAEI)[1] organizes the evaluation of ontology matching systems [162] and have obtained experiences in terms of the performance and matching strategies of ontology alignment systems (e.g., results in [163, 164, 165, 166, 167, 168, 169]), user validation in ontology alignment (e.g., [170, 171]) and complex ontology alignment (e.g., [172, 173]) which can be employed to the materials science field.

[1]http://oaei.ontologymatching.org/

Chapter

11

Conclusions

"I think you get more prestige by doing good science than by doing popular science because if you go with what you really think is important then it's a higher chance that it really is important in the long run and it's the long run which has the most benefit to the world."

Donald Knuth

We have now presented our solutions to the research questions and all the contributions of this thesis. In this chapter, we revisit the research questions and conclude this thesis. The goal of this thesis is to answer the following research question:

> *How to provide semantics-aware data access and data integration over heterogeneous data, following different models, being shared and queried via different ways?*

This question is further formulated into three sub-questions:

- **RQ1**: How can the recently developed GraphQL be used for semantics-aware data access and data integration over heterogeneous data sources?
- **RQ2**: How can ontologies be leveraged to generate GraphQL APIs for semantics-aware data access and data integration?
- **RQ3**: How can domain ontologies be extended by mining unstructured text, with validation from domain experts?

11.1 Ontology-driven data access and integration

In order to answer the first research question (**RQ1**), a GraphQL-based framework for data access and data integration was proposed. This framework contains two processes, which are the GraphQL server generation process and the GraphQL query answering process. The first process has to do with constructing GraphQL servers for the purpose of semantics-aware data access and data integration. We formulated the second research question (**RQ2**) concerning generation of GraphQL servers based on ontologies. Therefore, we proposed and implemented formal methods for generating GraphQL servers based on ontologies and semantic mappings. This process can be automated once suitable ontologies and semantic mappings have been defined. This automatic generation of GraphQL servers will help GraphQL application developers to avoid constructing every concrete detail of GraphQL servers. We developed a prototype (OBG-gen) to enable the automatic generation process. The second process is the normal query answering process in GraphQL applications, and the intended users are domain users who need to query data from different underlying data sources. The domain users may or may not have the background knowledge regarding the Semantic Web or ontologies. To write GraphQL queries, they need basic prior knowledge of GraphQL, which can be learned from the self-documenting API provided by the generated GraphQL server showing the schema.

11.2 Domain ontologies extension

It is sometimes necessary to add new databases or new types of data to existing databases in order to integrate data in a real-world application. Thus, the coverage of the ontology driving the GraphQL server generation may need to be enlarged. We studied how ontologies can be extended (**RQ3**) and proposed an approach (ToPMine-FTCA) based on phrase-based topic modeling, formal topical concept analysis and domain expert validation. The use of phrase-based topic modeling (ToPMine) aims at accomplishing the text mining task, and produces a list of frequent phrases and a list of latent topics, of which each topic contains a number of representative frequent phrases. Formal topical concept analysis over latent topics is intended to find relations among topics or phrases. In addition to the phrase-based topic modeling phase and the formal topical concept analysis phase, the approach includes a domain expert validation phase, during which a domain expert provides validations or interpretations of the results of the phrase-based topic modeling and the formal topical concept analysis. The validation or interpretation of such a concept or relation can serve as a basis for extending a domain ontology.

11.3 Evaluation and application in the materials science domain

As we conclude in Section 11.1 and Section 11.2, while solving the three research questions, we proposed the GraphQL-based framework for data access and integration, which contains a prototype (OBG-gen) implementation for automatic generation of a GraphQL server, and proposed an approach (ToPMine-FTCA) for extension of domain ontologies. In order to evaluate and apply the GraphQL-based framework and ToPMine-FTCA, we focused on the materials science field. This thesis is also based on a part of the project, SeRC-DCMD (*Swedish eScience Research Centre-Data Driven Computational Materials Design*), and is inspired by the work in the OPTIMADE consortium (*Open Databases Integration for Materials Design*). Therefore, we developed a domain ontology, the Materials Design Ontology (MDO), which is the first domain ontology for the materials design field. To design this ontology, we followed the best practices with respect to ontology engineering methodology. In the following steps, we first employed this ontology in the GraphQL-based framework and conducted experiments over a dataset based on two databases (Materials Project and OQMD) in the materials design field. Additionally, we discussed an application of this GraphQL-based framework and MDO within OPTIMADE. To evaluate and apply ToPMine-FTCA, we used it to extend two ontologies in the nanotechnology domain as well as to extend MDO.

There is a clear interest among materials scientists in making data FAIR, and recently there has been a lot of interest in Semantic Web-based technologies, but there has not been much practical application so far. Our contributions, in terms of MDO and ToPMine-FTCA, have been presented at a number of events in materials science (i.e., FAIR Data Infrastructure for Materials Genomics,[1] European Materials Modelling Council (EMMC) Multiscale Modelling of Materials and Molecules,[2] CECAM Open Databases Integration

[1]`https://th.fhi-berlin.mpg.de/meetings/fairdi2020/`
[2]`https://sites.google.com/site/emultiscale2020/`

for Materials Design[3] and Workshop on Ontologies for Materials-Databases Interoperability 2021[4]), and have attracted much interest.

[3]https://www.cecam.org/workshop-details/991

[4]https://www.optimade.org/omdi2021/

Bibliography

[1] Patrick Lambrix, Rickard Armiento, Anna Delin, and Huanyu Li. "Big Semantic Data Processing in the Materials Design Domain." In: *Encyclopedia of Big Data Technologies*. Springer, 2019. DOI: 10.1007/978-3-319-63962-8_293-1.

[2] Patrick Lambrix, Rickard Armiento, Anna Delin, and Huanyu Li. "FAIR Big Data in the Materials Design Domain." In: *Encyclopedia of Big Data Technologies*. accepted. Springer, 2022.

[3] Huanyu Li, Rickard Armiento, and Patrick Lambrix. "An Ontology for the Materials Design Domain." In: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020*. Vol. 12507. Lecture Notes in Computer Science. Springer, Cham, 2020, pp. 212–227. DOI: 10.1007/978-3-030-62466-8_14.

[4] Huanyu Li, Rickard Armiento, and Patrick Lambrix. "A Method for Extending Ontologies with Application to the Materials Science Domain." In: *Data Science Journal* 18.1 (2019). DOI: 10.5334/dsj-2019-050.

[5] Mina Abd Nikooie Pour, Huanyu Li, Rickard Armiento, and Patrick Lambrix. "A First Step towards Extending the Materials Design Ontology." In: *Proceedings of the Workshop on Domain Ontologies for Research Data Management in Industry Commons of Materials and

Manufacturing (DORIC-MM 2021) co-located with the 18th European Semantic Web Conference (ESWC 2021). 2021, pp. 1–11. URL: http://purl.org/net/epubs/work/50300311.

[6] Huanyu Li, Rickard Armiento, and Patrick Lambrix. "Extending Ontologies in the Nanotechnology Domain using Topic Models and Formal Topical Concept Analysis on Unstructured Text." In: *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019)*. Vol. 2456. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 5–8. URL: http://ceur-ws.org/Vol-2456/paper2.pdf.

[7] Tim Berners-Lee, James Hendler, and Ora Lassila. "THE SEMANTIC WEB." In: *Scientific American* 284.5 (2001), pp. 34–43. URL: http://www.jstor.org/stable/26059207.

[8] Dean Allemang, Jams A Hendler, and Fabien Gandon. *Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL*. 3rd ed. Association for Computing Machinery, 2020. DOI: 10.1145/3382097.

[9] John Domingue, Dieter Fensel, and James A Hendler. *Handbook of Semantic Web Technologies*. Springer, Berlin, Heidelberg, 2011. DOI: 10.1007/978-3-540-92913-0.

[10] Ankit Agrawal and Alok Choudhary. "Perspective: Materials informatics and big data: Realization of the "fourth paradigm" of science in materials science." In: *APL Materials* 4 (5 2016), 053208:1–10. DOI: 10.1063/1.4946894.

[11] Surya R Kalidindi and Marc De Graef. "Materials Data Science: Current Status and Future Outlook." In: *Annual Review of Materials Research* 45 (2015), pp. 171–193. DOI: 10.1146/annurev-matsci-070214-020844.

[12] Alexander Tropsha, Karmann C Mills, and Anthony J Hickey. "Reproducibility, sharing and progress in nanomaterial databases." In: *Nature Nanotechnology* 12 (2017), pp. 1111–1114. DOI: 10.1038/nnano.2017.233.

[13] Sandra Karcher, Egon L. Willighagen, John Rumble, Friederike Ehrhart, Chris T. Evelo, Martin Fritts, Sharon Gaheen, Stacey L. Harper, Mark D. Hoover, Nina Jeliazkova, Nastassja Lewinski, Richard L. Marchese Robinson, Karmann C. Mills, Axel P. Mustad, Dennis G. Thomas, Georgia Tsiliki, and Christine Ogilvie Hendren. "Integration among databases and data sets to support productive nanotechnology: Challenges and recommendations." In: *NanoImpact* 9 (2018), pp. 85–101. DOI: `10.1016/j.impact.2017.11.002`.

[14] John Rumble, John Broome, and Simon Hodson. "Building an International Consensus on Multi-Disciplinary Metadata Standards: A CODATA Case History in Nanotechnology." In: *Data Science Journal* 8 (2019), 12:1–11. DOI: `10.5334/dsj-2019-012`.

[15] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin a. Persson. "The Materials Project: A materials genome approach to accelerating materials innovation." In: *APL Materials* 1.1 (2013), p. 011002. DOI: `10.1063/1.4812323`.

[16] *The Materials Project*. `https://materialsproject.org`. Accessed: 2022-02-04.

[17] James E. Saal, Scott Kirklin, Muratahan Aykol, Bryce Meredig, and C. Wolverton. "Materials Design and Discovery with High-Throughput Density Functional Theory: The Open Quantum Materials Database (OQMD)." In: *JOM, The Journal of The Minerals, Metals & Materials Society (TMS)* 65.11 (Nov. 2013), pp. 1501–1509. DOI: `10.1007/s11837-013-0755-4`.

[18] *The Open Quantum Materials Database (OQMD)*. `http://oqmd.org`. Accessed: 2022-02-04.

[19] Claudia Draxl and Matthias Scheffler. "NOMAD: The FAIR concept for big data-driven materials science." In: *MRS Bulletin* 43.9 (2018), pp. 676–682. DOI: `10.1557/mrs.2018.208`.

[20] *Novel Materials Discovery (NOMAD)*. `https://repository.nomad-coe.eu/`. Accessed: 2022-02-04.

[21] Patrick Lambrix. “Towards a semantic Web for bioinformatics using ontology-based annotation.” In: *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE’05)*. 2005, pp. 3–7. DOI: `10.1109/WETICE.2005.58`.

[22] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C ’ t Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. “The FAIR Guiding Principles for scientific data management and stewardship.” In: *Scientific data* 3 (2016), 160018:1–9. DOI: `10.1038/sdata.2016.18`.

[23] Inc. Facebook. *Specification for GraphQL-June 2021 Edition*. `https://spec.graphql.org/October2021/`. Accessed: 2022-03-30. 2021.

[24] Qiang Liu and Patrick Lambrix. “A System for Debugging Missing Is-a Structure in Networked Ontologies.” In: *Data Integration in the Life Sciences, 7th International Conference, DILS 2010*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2010, pp. 50–57. DOI: `10.1007/978-3-642-15120-0_5`.

[25] Freddy Priyatna, Oscar Corcho, and Juan Sequeda. “Formalisation and Experiences of R2RML-Based SPARQL to SQL Query Translation Using Morph.” In: *Proceedings of the 23rd International Conference on World Wide Web*. Association for Computing Machinery, 2014, pp. 479–490. DOI: `10.1145/2566486.2567981`.

[26] David Chaves-Fraga, Edna Ruckhaus, Freddy Priyatna, Maria-Esther Vidal, and Oscar Corcho. “Enhancing Virtual Ontology Based Access over Tabular Data with Morph-CSV.” In: *Semantic Web* 12.6 (2021). DOI: `10.3233/SW-210432`.

[27] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. "Ontop: Answering SPARQL Queries Over Relational Databases." In: *Semantic Web* 8.3 (2017), pp. 471–487. DOI: 10.3233/SW-160217.

[28] Barbara Kitchenham. "Procedures for performing systematic reviews." In: *Keele, UK, Keele University* 33.2004 (2004), pp. 1–26.

[29] Joseph A Maxwell. *Qualitative research design: An interactive approach (3rd ed.)* SAGE Publications, Inc, 2012.

[30] Sharon M Ravitch and Matthew Riggan. *Reason & rigor: How conceptual frameworks guide research.* Sage Publications, Inc, 2016.

[31] Virginia Braun and Victoria Clarke. *SUCCESSFUL QUALITATIVE RESEARCH: a practical guide for beginners.* SAGE Publications Inc., 2013.

[32] Barry Smith. "Ontology." In: *The furniture of the world.* Brill, 2012, pp. 47–68. DOI: 10.1163/9789401207799_005.

[33] Christopher Welty. "Ontology Research." In: *AI Magazine* 24.3 (2003), p. 11. DOI: 10.1609/aimag.v24i3.1714.

[34] James H. Alexander, Michael J. Freiling, Sheryl J. Shulman, Jeffrey L. Staley, Steven Rehfuss, and Steven L. Messick. "Knowledge Level Engineering: Ontological Analysis." In: *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence.* AAAI'86. AAAI Press, 1986, pp. 963–967. URL: https://www.aaai.org/Papers/AAAI/1986/AAAI86-159.pdf.

[35] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. "Knowledge engineering: Principles and methods." In: *Data & Knowledge Engineering* 25.1 (1998), pp. 161–197. DOI: 10.1016/S0169-023X(97)00056-6.

[36] Robert Stevens, Carole A. Goble, and Sean Bechhofer. "Ontology-based Knowledge Representation for Bioinformatics." In: *Briefings in Bioinformatics* 1.4 (2000), pp. 398–414. DOI: 10.1093/bib/1.4.398.

[37] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications.* 2nd ed. Cambridge University Press, 2007. DOI: 10.1017/CBO9780511711787.

[38] *RDF 1.1 Concepts and Abstract Syntax W3C Recommendation 25 February 2014*. Technical Report. 2014. URL: https://www.w3.org/TR/rdf11-concepts/.

[39] *SPARQL 1.1 Overview W3C Recommendation 21 March 2013*. Technical Report. 2013. URL: https://www.w3.org/TR/sparql11-overview/.

[40] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. "Semantics and Complexity of SPARQL." In: *ACM Trans. Database Syst.* 34.3 (2009). DOI: 10.1145/1567274.1567278.

[41] Maurizio Lenzerini. "Data Integration: A Theoretical Perspective." In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. PODS '02. Association for Computing Machinery, 2002, pp. 233–246. DOI: 10.1145/543613.543644.

[42] Diego Calvanese and Giuseppe De Giacomo. "Data Integration: A Logic-Based Perspective." In: *AI magazine* 26.1 (2005), pp. 59–59. DOI: 10.1609/aimag.v26i1.1799.

[43] AnHai Doan, Alon Halevy, and Zachary Ives. "1 - Introduction." In: *Principles of Data Integration*. Morgan Kaufmann, 2012, pp. 1–18. DOI: 10.1016/B978-0-12-416044-6.00001-6.

[44] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "Ontology-Based Data Access and Integration." In: *Encyclopedia of Database Systems*. Springer, New York, NY, 2018, pp. 2590–2596. DOI: 10.1007/978-1-4614-8265-9_80667.

[45] Guohui Xiao, Dag Hovland, Dimitris Bilidas, Martin Rezk, Martin Giese, and Diego Calvanese. "Efficient Ontology-Based Data Integration with Canonical IRIs." In: *The Semantic Web 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings*. Vol. 10843. Lecture Notes in Computer Science. Springer,Cham, 2018, pp. 697–713. DOI: 10.1007/978-3-319-93417-4_45.

[46] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. "Ontology-Based Data Access: A Survey." In: *Proceedings of the*

Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 5511–5519. DOI: 10.24963/ijcai.2018/777.

[47] Oscar Corcho, Freddy Priyatna, and David Chaves-Fraga. "Towards a new generation of ontology based data access." In: *Semantic Web* 11.1 (2020), pp. 153–160. DOI: 10.3233/SW-190384.

[48] Gio Wiederhold. "Mediators in the Architecture of Future Information Systems." In: *Computer* 25.3 (1992), pp. 38–49. DOI: 10.1109/2.121508.

[49] Panos Vassiliadis. "A Survey of Extract–Transform–Load Technology." In: *Integrations of Data Warehousing, Data Mining and Database Technologies: Innovative Approaches* 5.3 (2009), pp. 1–27. DOI: 10.4018/978-1-60960-537-7.ch008.

[50] Souripriya Das, Seema Sundara, and Richard Cyganiak. *R2RML: RDB to RDF Mapping Language*. https://www.w3.org/TR/r2rml/. Accessed: 2022-02-04.

[51] Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, and Juan Sequeda. *A Direct Mapping of Relational Data to RDF*. https://www.w3.org/TR/rdb-direct-mapping/. Accessed: 2022-02-04.

[52] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data." In: *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014)*. Vol. 1184. CEUR Workshop Proceedings. CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[53] Anastasia Dimou, Miel Vander Sande, Jason Slepicka, Pedro Szekely, Erik Mannens, Craig Knoblock, and Rik Van de Walle. "Mapping Hierarchical Sources into RDF Using the RML Mapping Language." In: *2014 IEEE International Conference on Semantic Computing*. IEEE, 2014, pp. 151–158. DOI: 10.1109/ICSC.2014.25.

[54] European Commission, Entrepreneurship Directorate-General for Internal Market Industry, SMEs, S Bobba, P Claudiu, D Huygens, P Alves Dias, B Gawlik, E Tzimas, D Wittmer, P Nuss, M Grohol, H Saveyn, F Buraoui, G Orveillon, T Hámor, S Slavko, F Mathieux, M Gislev, C Torres De Matos, G Blengini, F Ardente, D Blagoeva, and E Garbarino. *Report on critical raw materials and the circular economy.* Publications Office, 2018. DOI: `10.2873/331561`.

[55] Kurt Lejaeghere, Gustav Bihlmayer, Torbjörn Björkman, Peter Blaha, Stefan Blügel, Volker Blum, Damien Caliste, Ivano E. Castelli, Stewart J. Clark, Andrea Dal Corso, Stefano de Gironcoli, Thierry Deutsch, John Kay Dewhurst, Igor Di Marco, Claudia Draxl, Marcin Dulak, Olle Eriksson, Jose A. Flores-Livas, Kevin F. Garrity, Luigi Genovese, Paolo Giannozzi, Matteo Giantomassi, Stefan Goedecker, Xavier Gonze, Oscar Grånäs, E. K. U. Gross, Andris Gulans, Francois Gygi, D. R. Hamann, Phil J. Hasnip, N. A. W. Holzwarth, Diana Iusan, Dominik B. Jochym, François Jollet, Daniel Jones, Georg Kresse, Klaus Koepernik, Emine Kücükbenli, Yaroslav O. Kvashnin, Inka L. M. Locht, Sven Lubeck, Martijn Marsman, Nicola Marzari, Ulrike Nitzsche, Lars Nordström, Taisuke Ozaki, Lorenzo Paulatto, Chris J. Pickard, Ward Poelmans, Matt I. J. Probert, Keith Refson, Manuel Richter, Gian-Marco Rignanese, Santanu Saha, Matthias Scheffler, Martin Schlipf, Karlheinz Schwarz, Sangeeta Sharma, Francesca Tavazza, Patrik Thunström, Alexandre Tkatchenko, Marc Torrent, David Vanderbilt, Michiel J. van Setten, Veronique Van Speybroeck, John M. Wills, Jonathan R. Yates, Guo-Xu Zhang, and Stefaan Cottenier. "Reproducibility in density functional theory calculations of solids." In: *Science* 351.6280 (2016), aad3000. DOI: `10.1126/science.aad3000`.

[56] Rickard Armiento. "Database-Driven High-Throughput Calculations and Machine Learning Models for Materials Design." In: *Machine Learning Meets Quantum Physics.* Vol. 968. Lecture Notes in Physics. Springer, Cham, 2020. DOI: `10.1007/978-3-030-40245-7_17`.

[57] Cleidson R. B. de Souza, David Redmiles, Li-Te Cheng, David Millen, and John Patterson. "Sometimes You Need to See through Walls: A Field Study of Application Programming Interfaces." In: *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work.*

CSCW '04. Association for Computing Machinery, 2004, pp. 63–71. DOI: 10.1145/1031607.1031620.

[58] Oscar Borgogno and Giuseppe Colangelo. "Data sharing and interoperability: Fostering innovation and competition through APIs." In: *Computer Law & Security Review* 35.5 (2019), p. 105314. DOI: 10.1016/j.clsr.2019.03.008.

[59] Milan Dojchinovski and Tomas Vitvar. "Linked Web APIs dataset." In: *Semantic Web* 9.4 (2018), pp. 381–391. DOI: 10.3233/SW-170259.

[60] Diego Serrano, Eleni Stroulia, Diana Lau, and Tinny Ng. "Linked REST APIs: A Middleware for Semantic REST API Integration." In: *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 138–145. DOI: 10.1109/ICWS.2017.26.

[61] Simon J. D. Cox, Alejandra N. Gonzalez-Beltran, Barbara Magagna, and Maria-Cristina Marinescu. "Ten simple rules for making a vocabulary FAIR." In: *PLOS Computational Biology* 17.6 (2021), pp. 1–15. DOI: 10.1371/journal.pcbi.1009041.

[62] Daniel Garijo and Maria Poveda-Villalón. "Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web." In: *Applications and Practices in Ontology Design, Extraction, and Reasoning*. IOS Press, 2020. DOI: 10.3233/SSW200034.

[63] Olaf Hartig and Jan Hidders. "Defining Schemas for Property Graphs by Using the GraphQL Schema Definition Language." In: *Proceedings of the 2nd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*. GRADES-NDA'19. Association for Computing Machinery, 2019. DOI: 10.1145/3327964.3328495.

[64] Olaf Hartig and Jorge Pérez. "Semantics and Complexity of GraphQL." In: *Proceedings of the 2018 World Wide Web Conference*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 1155–1164. DOI: 10.1145/3178876.3186014.

[65] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. 1st. USA: Cambridge University Press, 2017. ISBN: 0521695422. DOI: 10.1017/9781139025355.

[66] Franz Baader, Pavlos Marantidis, and Maximilian Pensel. "The Data Complexity of Answering Instance Queries in FL0." In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. International World Wide Web Conferences Steering Committee, 2018, pp. 1603–1607. DOI: 10.1145/3184558.3191618.

[67] Thomas M. Connolly and Carolyn E. Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 5th ed. PEARSON Education, 2010.

[68] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. "Linking Data to Ontologies." In: *Journal on Data Semantics X*. Springer, Berlin, Heidelberg, 2008, pp. 133–173. DOI: 10.1007/978-3-540-77688-8_5.

[69] David Chaves-Fraga, Freddy Priyatna, Ahmad Alobaid, and Oscar Corcho. "Exploiting Declarative Mapping Rules for Generating GraphQL Servers with Morph-GraphQL." In: *International Journal of Software Engineering and Knowledge Engineering* 30.06 (2020), pp. 785–803. DOI: 10.1142/S0218194020400070.

[70] *morph-rdb, version 3.12.5*. https://github.com/oeg-upm/morph-rdb/releases/tag/v3.12.5. Accessed: 2022-02-04.

[71] Ruben Taelman, Miel Vander Sande, and Ruben Verborgh. "GraphQL-LD: Linked Data Querying with GraphQL." In: *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018)*. Vol. 2180. CEUR Workshop Proceedings. CEUR-WS, 2018. URL: http://ceur-ws.org/Vol-2180/paper-65.pdf.

[72] Semantic Integration Ltd. *HyperGraphQL, version 2.0.0*. https://github.com/hypergraphql/hypergraphql/releases/tag/2.0.0. Accessed: 2022-02-04.

[73] Lars Gleim, Tim Holzheim, István Koren, and Stefan Decker. "Automatic Bootstrapping of GraphQL Endpoints for RDF Triple Stores." In: *Joint Proceedings of Workshops AI4LEGAL2020, NLIWOD, PROFILES 2020, QuWeDa 2020 and SEMIFORM2020 co-located with the 19th International Semantic Web Conference (ISWC 2020)*. Vol. 2722. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 119–134. URL: http://ceur-ws.org/Vol-2722/quweda2020-paper-2.pdf.

[74] Semantic Integration Ltd. *UltraGraphQL, version 1.0.0.* `https://git.rwth-aachen.de/i5/ultragraphql`. Accessed: 2022-02-04.

[75] Daniel Garijo and Maximiliano Osorio. “OBA: An Ontology-Based Framework for Creating REST APIs for Knowledge Graphs.” In: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020.* Vol. 12507. Lecture Notes in Computer Science. Springer, Cham, 2020, pp. 48–64. DOI: `10.1007/978-3-030-62466-8_4`.

[76] Carles Farré, Jovan Varga, and Robert Almar. “GraphQL Schema Generation for Data-Intensive Web APIs.” In: *Model and Data Engineering.* Springer, Cham, 2019, pp. 184–194. DOI: `10.1007/978-3-030-32065-2_13`.

[77] York Sure, Steffen Staab, and Rudi Studer. “Ontology Engineering Methodology.” In: *Handbook on Ontologies.* Springer, Berlin, Heidelberg, 2009, pp. 135–152. DOI: `10.1007/978-3-540-92673-3_6`.

[78] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. “OOPS! (OntOlogy Pitfall Scanner!): An On-Line Tool for Ontology Evaluation.” In: 10.2 (2014). DOI: `10.4018/ijswis.2014040102`.

[79] Patrick Lambrix. *Completing and Debugging Ontologies: state of the art and challenges.* arXiv:1908.03171. 2020.

[80] Ahmad Alobaid, Daniel Garijo, María Poveda-Villalón, Idafen Santana-Pérez, and Óscar Corcho. “OnToology, a tool for collaborative development of ontologies.” In: *Proceedings of the International Conference on Biomedical Ontology.* Vol. 1515. CEUR Workshop Proceedings. CEUR-WS.org, 2015. URL: `http://ceur-ws.org/Vol-1515/demo3.pdf`.

[81] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. “METHONTOLOGY: from Ontological Art towards Ontological Engineering.” In: *Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series.* 1997, pp. 33–40. URL: `https://www.aaai.org/Papers/Symposia/Spring/1997/SS-97-06/SS97-06-005.pdf`.

[82] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. "The NeOn Methodology for Ontology Engineering." In: *Ontology Engineering in a Networked World*. Springer, Berlin, Heidelberg, 2012, pp. 9–34. DOI: `10.1007/978-3-642-24794-1_2`.

[83] María Poveda-Villalón. "A Reuse-Based Lightweight Method for Developing Linked Data Ontologies and Vocabularies." In: *The Semantic Web: Research and Applications*. Springer, Berlin, Heidelberg, 2012, pp. 833–837. DOI: `10.1007/978-3-642-30284-8_66`.

[84] Raúl García-Castro, Alba Fernández-Izquierdo, Christopher Heinz, Peter Kostelnik, María Poveda-Villalón, and Fernando Serena. *D2.2 Detailed specification of the semantic model.* Technical Report. 2017. URL: `https://vicinity2020.eu/vicinity/content/d22-detailed-specification-semantic-model`.

[85] York Sure, Steffen Staab, and Rudi Studer. "On-To-Knowledge Methodology (OTKM)." In: *Handbook on Ontologies*. Ed. by Steffen Staab and Rudi Studer. Springer, Berlin, Heidelberg, 2004, pp. 117–132. DOI: `10.1007/978-3-540-24750-0_6`.

[86] Valentina Presutti, Eva Blomqvist, Enrico Daga, and Aldo Gangemi. "Pattern-Based Ontology Design." In: *Ontology Engineering in a Networked World*. Ed. by Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi. Springer, Berlin, Heidelberg, 2012, pp. 35–64. DOI: `10.1007/978-3-642-24794-1_3`.

[87] Zlatan Dragisic, Patrick Lambrix, and Eva Blomqvist. "Integrating Ontology Debugging and Matching into the eXtreme Design Methodology." In: *Proceedings of the 6th Workshop on Ontology and Semantic Web Patterns (WOP 2015) co-located with the 14th International Semantic Web Conference (ISWC 2015)*. Vol. 1461. CEUR Workshop Proceedings. CEUR-WS.org, 2015. URL: `http://ceur-ws.org/Vol-1461/WOP2015_paper_1.pdf`.

[88] Kwok Cheung, John Drennan, and Jane Hunter. "Towards an Ontology for Data-driven Discovery of New Materials." In: *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*. 2008, pp. 9–14. URL: `https://www.aaai.org/Papers/Symposia/Spring/2008/SS-08-05/SS08-05-003.pdf`.

[89] Xiaoming Zhang, Changjun Hu, and Huayu Li. “Semantic query on materials data based on mapping MATML to an OWL ontology.” In: *Data Science Journal* 8 (2009), pp. 1–17. DOI: `10.2481/dsj.8.1`.

[90] Toshihiro Ashino. “Materials Ontology: An Infrastructure for Exchanging Materials Information and Knowledge.” In: *Data Science Journal* 9 (2010), pp. 54–61. DOI: `10.2481/dsj.008-041`.

[91] Dennis G Thomas, Rohit V Pappu, and Nathan A Baker. “NanoParticle Ontology for cancer nanotechnology research.” In: *Journal of Biomedical Informatics* 44.1 (2011), pp. 59–74. DOI: `10.1016/j.jbi.2010.03.001`.

[92] Robert Arp, Barry Smith, and Andrew D. Spear. *Building Ontologies with Basic Formal Ontology.* The MIT Press, 2015. URL: `https://mitpress.mit.edu/books/building-ontologies-basic-formal-ontology`.

[93] Janna Hastings, Nina Jeliazkova, Gareth Owen, Georgia Tsiliki, Cristian R Munteanu, Christoph Steinbeck, and Egon Willighagen. “eNanoMapper: harnessing ontologies to enable data integration for nanomaterial risk assessment.” In: *Journal of Biomedical Semantics* 6.1 (2015), p. 10. DOI: `10.1186/s13326-015-0005-5`.

[94] Xiaoming Zhang, Dongyu Pan, Chongchong Zhao, and Kai Li. “MMOY: Towards deriving a metallic materials ontology from Yago.” In: *Advanced Engineering Informatics* 30 (2016), pp. 687–702. DOI: `10.1016/j.aei.2016.09.002`.

[95] Fabio ALe Piane, Matteo Baldoni, Mauro GCaspari, and Francesco Merucuri. “Introducing MAMBO: Materials And Molecules Basic Ontology.” In: *Proceedings of the Workshop on Domain Ontologies for Research Data Management in Industry Commons of Materials and Manufacturing (DORIC-MM 2021) co-located with the 18th European Semantic Web Conference (ESWC 2021).* 2021, pp. 28–39. URL: `http://purl.org/net/epubs/work/50300311`.

[96] Ahmad Zainul Ihsan, Danilo Dessì, Mehwish Alam, Harald Sack, and Stefan Sandfeld. “Steps towards a Dislocation Ontology for Crystalline Materials.” In: *Proceedings of the Second International Workshop on Semantic Digital Twins co-located with the 18th Extended Semantic Web Conference (ESWC 2021).* Vol. 2887. CEUR Workshop Proceed-

ings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2887/paper4.pdf.

[97] Mehwish Alam, Henk Birkholz, Danilo Dessì, Christoph Eberl, Heike Fliegl, Peter Gumbsch, Philipp von Hartrott, Lutz Mädler, Markus Niebel, Harald Sack, and Akhil Thomas. "Ontology Modelling for Materials Science Experiments." In: *Proceedings of the Poster & Demo track co-located with he 17th International Conference on Semantic Systems (SEMANTiCS 2021).* Vol. 2941. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2941/paper11.pdf.

[98] European Committee for Standardization (CEN). "A Guide to the Development and Use of Standards Compliant Data Formats for Engineering Materials Test Data." In: (2010). European Committee for standardization. URL: https://joinup.ec.europa.eu/collection/european-committee-standardization-cen/solution/guide-development-and-use-standards-compliant-data-formats-engineering-materials-test-data/about.

[99] *Inorganic Crystal Structure Database (ICSD).* https://icsd.fiz-karlsruhe.de. Accessed: 2022-02-04.

[100] Alec Belsky, Mariette Hellenbrandt, Vicky Lynn Karen, and Peter Luksch. "New developments in the Inorganic Crystal Structure Database (ICSD): accessibility in support of materials research and design." In: *Acta Crystallographica Section B: Structural Science* 58.3 (2002), pp. 364–369. DOI: 10.1107/S0108768102006948.

[101] G. Bergerhoff, R. Hundt, R. Sievers, and I. D. Brown. "The inorganic crystal structure data base." In: *Journal of Chemical Information and Computer Sciences* 23.2 (1983), pp. 66–69. DOI: 10.1021/ci00038a003.

[102] Leslie Glasser. "Crystallographic Information Resources." In: *Journal of Chemical Education* 93 (2016), pp. 542–549. DOI: 10.1021/acs.jchemed.5b00253.

[103] *Crystallography Open Database (COD).* http://www.crystallography.net/cod/. Accessed: 2022-02-04.

[104] Saulius Grazulis, Adriana Dazkevic, Andrius Merkys, Daniel Chateigner, Luca Lutterotti, Miguel Quiros, Nadezhda R. Serebryanaya, Peter Moeck, Robert T. Downs, and Armel Le Bail. "Crystallography Open Database (COD): an open-access collection of crystal structures and platform for world-wide collaboration." In: *Nucleic Acids Research* 40.Database issue (2012), pp. D420–D427. DOI: `10.1093/nar/gkr900`.

[105] *Predicted Crystallography Open Database (PCOD).* `http://www.crystallography.net/pcod/`. Accessed: 2022-02-04.

[106] *Theoretical Crystallography Open Database (TCOD).* `http://www.crystallography.net/tcod/`. Accessed: 2022-02-04.

[107] *The International Centre for Diffraction Data (ICDD).* `https://www.icdd.com`. Accessed: 2022-02-04.

[108] *Springer Materials.* `https://materials.springer.com`. Accessed: 2022-02-04.

[109] *The National Institute for Materials Science (NIMS) Materials Database (MatNavi).* `https://www.nims.go.jp/eng/`. Accessed: 2022-02-04.

[110] C. E. Campbell, U. R. Kattner, and Z.-K. Liu. "File and data repositories for Next Generation CALPHAD." In: *Scripta Materialia* 70.Supplement C (2014), pp. 7–11. DOI: `10.1016/j.scriptamat.2013.06.013`.

[111] *The databases provided by OpenCalhad.* `http://www.opencalphad.com/databases.html`. Accessed: 2022-02-04.

[112] V. L. Moruzzi, J. F. Janak, and A. R. Williams. *Calculated Electronic Properties of Metals.* Pergamon Press, 2013. DOI: `10.1016/C2013-0-03017-4`.

[113] *The Electronic Structure Project (ESP).* `http://materialsgenome.se`. Accessed: 2022-02-04.

[114] Stefano Curtarolo, Wahyu Setyawan, Shidong Wang, Junkai Xue, Kesong Yang, Richard Taylor, Lance Nelson, Gus Hart, Stefano Sanvito, Marco Buongiorno-Nardelli, Natalio Mingo, and Ohad Levy. "AFLOWLIB.ORG: A distributed materials properties repository from

high-throughput ab initio calculations.” In: *Computational Materials Science* 58.Supplement C (2012), pp. 227–235. DOI: 10.1016/j.commatsci.2012.02.002.

[115] *Automatic Flow for Materials Discovery (AFLOW).* ttp://aflowlib.org/. Accessed: 2022-02-04.

[116] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. “Commentary: The Materials Project: A materials genome approach to accelerating materials innovation.” In: *APL Materials* 1.1 (2013), p. 011002. DOI: 10.1063/1.4812323.

[117] Giovanni Pizzi, Andrea Cepellotti, Riccardo Sabatini, Nicola Marzari, and Boris Kozinsky. “AiiDA: automated interactive infrastructure and database for computational science.” In: *Computational Materials Science* 111.Supplement C (2016), pp. 218–230. DOI: 10.1016/j.commatsci.2015.09.013.

[118] *Automated Interactive Infrastructure and Database for Computational Science (AiiDA).* https://www.aiida.net. Accessed: 2022-02-04.

[119] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E. Castelli, Rune Christensen, Marcin Dulak, Jesper Friis, Michael N. Groves, Bjørk Hammer, Cory Hargus, Eric D. Hermes, Paul C. Jennings, Peter Bjerre Jensen, James Kermode, John R. Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S. Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W. Jacobsen. “The atomic simulation environment - a Python library for working with atoms.” In: *Journal of Physics: Condensed Matter* 29.27 (2017), p. 273002. DOI: 10.1088/1361-648X/aa680e.

[120] *The Atomic Simulation Environment (ASE).* https://wiki.fysik.dtu.dk/ase/. Accessed: 2022-02-04.

[121] Felix Faber, Alexander Lindmaa, Anatole von Lilienfeld, and Rickard Armiento. “Machine Learning Energies of 2 Million Elpasolite

(ABC_2D_6) Crystals." In: *Physical Review Letters* 117.13 (Sept. 2016), p. 135502. DOI: 10.1103/PhysRevLett.117.135502.

[122] *The High-Throughput Toolkit (httk).* https://httk.org. Accessed: 2022-02-04.

[123] Casper W Andersen, Rickard Armiento, Evgeny Blokhin, Gareth J Conduit, Shyam Dwaraknath, Matthew L Evans, Ádám Fekete, Abhijith Gopakumar, Saulius Gražulis, Andrius Merkys, et al. "OPTIMADE: an API for exchanging materials data." In: *Scientific Data* 8.217 (2021). DOI: 10.1038/s41597-021-00974-z.

[124] Paula de Matos, Adriano Dekker, Marcus Ennis, Janna Hastings, Kenneth Haug, Steve Turner, and Christoph Steinbeck. "ChEBI: a chemistry ontology and database." In: *Journal of Cheminformatics* 2.P6 (2010). DOI: 10.1186/1758-2946-2-S1-P6.

[125] Ralph Haas, Paul J Keller, Jack Hodges, and Jack Spivak. *Quantities, units, dimensions and data types ontologies (QUDT).* http://qudt.org. Accessed: 2022-02-04.

[126] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. *PROV-O: The PROV Ontology.* https://www.w3.org/TR/prov-o/. Accessed: 2022-02-04. 2013.

[127] Paul Buitelaar, Phillip Cimiano, and Bernardo Magnini. *Ontology Learning from Text: Methods, Evaluation and Applications.* Vol. 123. IOS Press, 2005.

[128] Muhammad Nabeel Asim, Muhammad Wasim, Muhammad Usman Ghani Khan, Waqar Mahmood, and Hafiza Mahnoor Abbasi. "A survey of ontology learning techniques and applications." In: *Database* 2018 (2018), bay101:1–24. DOI: 10.1093/database/bay101.

[129] Marti A. Hearst. "Automatic acquisition of hyponyms from large text corpora." In: *14th International Conference on Computational Linguistics.* 1992, pp. 539–545. DOI: 10.3115/992133.992154.

[130] Tomas Wächter, He Tan, Andre Wobst, Patrick Lambrix, and Michael Schroeder. "A Corpus-Driven Approach for Design, Evolution and Alignment of Ontologies." In: *Proceedings of the 2006 Winter Simulation Conference.* IEEE, 2006, pp. 1595–1602. DOI: 10.1109/WSC.2006.322932.

[131] Patrick Arnold and Erhard Rahm. "Semantic Enrichment of Ontology Mappings: A Linguistic-Based Approach." In: *17th East European Conference on Advances in Databases and Information Systems*. Vol. 8133. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2013, pp. 42–55. DOI: 10.1007/978-3-642-40683-6_4.

[132] Alexander Maedche, Viktor Pekar, and Steffen Staab. "Ontology Learning Part One — On Discovering Taxonomic Relations from the Web." In: *Web Intelligence*. Springer, Berlin, Heidelberg, 2003, pp. 301–320. DOI: 10.1007/978-3-662-05320-1_14.

[133] Alexander Maedche and Steffen Staab. "Discovering Conceptual Relations from Text." In: *Proceedings of the 14th European Conference on Artificial Intelligence*. ECAI'00. IOS Press, 2000, pp. 321–325. URL: https://dl.acm.org/doi/10.5555/3006433.3006501.

[134] Elias Zavitsanos, Georgios Paliouras, George A. Vouros, and Sergios Petridis. "Discovering Subsumption Hierarchies of Ontology Concepts from Text Corpora." In: *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*. IEEE, 2007, pp. 402–408. DOI: 10.1109/WI.2007.55.

[135] Vassilis Spiliopoulos, George A.Vouros, and Vangelis Karkaletsis. "On the discovery of subsumption relations for the alignment of ontologies." In: *Journal of Web Semantics* 8 (2010), pp. 69–88. DOI: 10.1016/j.websem.2010.01.001.

[136] Phillip Cimiano, Andreas Hotho, and Steffen Staab. "Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis." In: *Journal of Artificial Intelligence Research* 24 (2005), pp. 305–339. DOI: 10.1613/jair.1648.

[137] Markus Schaal, Roland M. Müller, Marko Brunzel, and Myra Spiliopoulou. "RELFIN - Topic Discovery for Ontology Enhancement and Annotation." In: *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*. 2005, pp. 608–622. DOI: 10.1007/11431053_41.

[138] Zhijie Lin, Rui Lu, Yun Xiong, and Yangyong Zhu. "Learning Ontology Automatically Using Topic Model." In: *2012 International Conference*

on Biomedical Engineering and Biotechnology. IEEE, 2012, pp. 360–363. DOI: 10.1109/iCBEB.2012.263.

[139] Monika Rani, Amit Kumar Dhar, and O. P. Vyas. "Semi-automatic terminology ontology learning based on topic modeling." In: *Engineering Applications of Artificial Intelligence* 63 (2017), pp. 108–125. DOI: 10.1016/j.engappai.2017.05.006.

[140] Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R. Voss, and Jiawei Han. "Scalable Topical Phrase Mining from Text Corpora." In: *Proceedings of the VLDB Endowment* 8.3 (2014), pp. 305–316. DOI: 10.14778/2735508.2735519.

[141] Michael Hartung, James Terwilliger, and Erhard Rahm. "Recent Advances in Schema and Ontology Evolution." In: *Schema Matching and Mapping*. Springer, Berlin, Heidelberg, 2011, pp. 149–190. DOI: 10.1007/978-3-642-16518-4_6.

[142] Julio Cesar Dos Reis, Duy Dinh, Cedric Pruski, Marcos Da Silveira, and Chantal Reynaud-Delaitre. "Mapping adaptation actions for the automatic reconciliation of dynamic ontologies." In: *22nd ACM International Conference on Information and Knowledge Management*. CIKM '13. Association for Computing Machinery, 2013, pp. 599–608. DOI: 10.1145/2505515.2505564.

[143] Valentina Ivanova and Patrick Lambrix. "A Unified Approach for Aligning Taxonomies and Debugging Taxonomies and Their Alignments." In: *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*. Vol. 7882. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2013, pp. 1–15. DOI: 10.1007/978-3-642-38288-8_1.

[144] Patrick Lambrix, Fang Wei-Kleiner, and Zlatan Dragisic. "Completing the is-a structure in light-weight ontologies." In: *Journal of Biomedical Semantics* 6 (2015), 12:1–26. DOI: 10.1186/s13326-015-0002-8.

[145] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[146] David M. Blei. "Probabilistic Topic Models." In: *Commun. ACM* 55.4 (2012), pp. 77–84. DOI: 10.1145/2133806.2133826.

[147] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis*. Springer, Berlin, Heidelberg, 1999. DOI: 10.1007/978-3-642-59830-2.

[148] David Faure and Thierry Poibeau. "First Experiments of Using Semantic Knowledge Learned by ASIUM for Information Extraction Task Using INTEX." In: *Proceedings of the First International Conference on Ontology Learning - Volume 31*. OL'00. CEUR-WS.org, 2000, pp. 7–12. URL: https://dl.acm.org/doi/10.5555/3053703.3053706.

[149] Xing Jiang and Ah-Hwee Tan. "CRCTOL: A semantic-based domain ontology learning system." In: *Journal of the American Society for Information Science and Technology* 61.1 (2010), pp. 150–168. DOI: 10.1002/asi.21231.

[150] Euthymios Drymonas, Kalliopi Zervanou, and Euripides G. M. Petrakis. "Unsupervised Ontology Acquisition from Plain Texts: The OntoGain System." In: *Natural Language Processing and Information Systems 15th International Conference on Applications of Natural Language to Information Systems, NLDB 2010, Cardiff, UK, June 23-25, 2010. Proceedings*. Vol. 6177. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2010, pp. 277–287. DOI: 10.1007/978-3-642-13881-2_29.

[151] Roberto Navigli, Paola Velardi, Alessandro Cucchiarelli, and Francesca Neri. "Extending and enriching WordNet with OntoLearn." In: *Proceedings of the 2nd Global WordNet Conference (GWC)*. 2004, pp. 279–284. URL: http://www.fi.muni.cz/gwc2004/proc/86.pdf.

[152] Philipp Cimiano and Johanna Völker. "Text2Onto." In: *Natural Language Processing and Information Systems 10th International Conference on Applications of Natural Language to Information Systems, NLDB 2005, Alicante, Spain, June 15-17, 2005. Proceedings*. Vol. 3513. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2005, pp. 227–238. DOI: 10.1007/11428817_21.

[153] Wilson Wong, Wei Liu, and Mohammed Bennamoun. "Ontology learning from text: A look back and into the future." In: *ACM Computing Surveys* 44.4 (2012), p. 20. DOI: 10.1145/2333112.2333115.

[154] Roberto Navigli and Paola Velardi. "Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites." In: *Compu-*

tational Linguistics 30.2 (June 2004), pp. 151–179. DOI: 10.1162/089120104323093276.

[155] National Institute for Occupational Safety and Health (NIOSH). *NanoParticle Information Library (NIF).* http://nanoparticlelibrary.net. Accessed: 2022-02-04.

[156] Lukas Galke, Florian Mai, Alan Schelten, Dennis Brunsch, and Ansgar Scherp. “Using Titles vs. Full-text as Source for Automated Semantic Document Annotation.” In: *Proceedings of the Knowledge Capture Conference.* K-CAP 2017. 2017, 20:1–4. DOI: 10.1145/3148011.3148039.

[157] Mark Steyvers and Tom Griffiths. “Probabilistic Topic Models.” In: *Latent Semantic Analysis: A Road to Meaning.* Laurence Erlbaum, 2007.

[158] James E. Saal, Scott Kirklin, Muratahan Aykol, Bryce Meredig, and C. Wolverton. “Materials Design and Discovery with High-Throughput Density Functional Theory: The Open Quantum Materials Database (OQMD).” In: *JOM, The Journal of The Minerals, Metals & Materials Society (TMS)* 65 (2013), pp. 1501–1509. DOI: 10.1007/s11837-013-0755-4.

[159] Matthew L Evans, Casper W Andersen, Shyam Dwaraknath, Markus Scheidgen, Ádám Fekete, and Donald Winston. “optimade-python-tools: a Python library for serving and consuming materials data via OPTIMADE APIs.” In: *Journal of Open Source Software* 6.65 (2021), p. 3458. DOI: 10.21105/joss.03458.

[160] Mina Abd Nikooie Pour, Huanyu Li, Rickard Armiento, and Patrick Lambrix. “A First Step towards a Tool for Extending Ontologies.” In: *Proceedings of the Sixth International Workshop on the Visualization and Interaction for Ontologies and Linked Data co-located with the 20th International Semantic Web Conference (ISWC 2021).* Vol. 3023. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 1–12. URL: http://ceur-ws.org/Vol-3023/paper2.pdf.

[161] Zlatan Dragisic, Valentina Ivanova, Huanyu Li, and Patrick Lambrix. “Experiences from the Anatomy track in the Ontology Alignment Evaluation Initiative.” In: *Journal of Biomedical Semantics* 8 (2017), 56:1–56:28. DOI: 10.1186/s13326-017-0166-5.

[162] Jérôme Euzenat, Christian Meilicke, Heiner Stuckenschmidt, Pavel Shvaiko, and Cássia Trojahn. "Ontology Alignment Evaluation Initiative: Six Years of Experience." In: *Journal on Data Semantics XV*. Springer, Berlin, Heidelberg, 2011, pp. 158–192. DOI: 10.1007/978-3-642-22630-4_6.

[163] Manel Achichi, Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Ian Harrow, Valentina Ivanova, Ernesto Jiménez-Ruiz, Elena Kuss, Patrick Lambrix, Henrik Leopold, Huanyu Li, Christian Meilicke, Stefano Montanelli, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Heiner Stuckenschmidt, Konstantin Todorov, Cássia Trojahn, and Ondrej Zamazal. "Results of the Ontology Alignment Evaluation Initiative 2016." In: *Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016)*. Vol. 1766. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 73–129. URL: http://ceur-ws.org/Vol-1766/oaei16_paper0.pdf.

[164] Manel Achichi, Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Ian Harrow, Valentina Ivanova, Ernesto Jiménez-Ruiz, Kristian Kolthoff, Elena Kuss, Patrick Lambrix, Henrik Leopold, Huanyu Li, Christian Meilicke, Majid Mohammadi, Stefano Montanelli, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Heiner Stuckenschmidt, Élodie Thiéblin, Konstantin Todorov, Cássia Trojahn, and Ondrej Zamazal. "Results of the Ontology Alignment Evaluation Initiative 2017." In: *Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017)*. Vol. 2032. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 61–113. URL: http://ceur-ws.org/Vol-2032/oaei17_paper0.pdf.

[165] Alsayed Algergawy, Michelle Cheatham, Daniel Faria, Alfio Ferrara, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Naouel Karam, Abderrahmane Khiat, Patrick Lambrix, Huanyu Li, Stefano Montanelli, Heiko Paulheim, Catia Pesquita, Tzanina Saveta, Daniela Schmidt, Pavel Shvaiko, Andrea Splendiani, Élodie Thiéblin, Cássia Trojahn, Jana Vatascinová, Ondrej Zamazal, and Lu Zhou. "Re-

sults of the Ontology Alignment Evaluation Initiative 2018." In: *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference (ISWC 2018)*. Vol. 2288. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 76–116. URL: http://ceur-ws.org/Vol-2288/oaei18_paper0.pdf.

[166] Alsayed Algergawy, Daniel Faria, Alfio Ferrara, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Naouel Karam, Abderrahmane Khiat, Patrick Lambrix, Huanyu Li, Stefano Montanelli, Heiko Paulheim, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Élodie Thiéblin, Cássia Trojahn, Jana Vatascinová, Ondrej Zamazal, and Lu Zhou. "Results of the Ontology Alignment Evaluation Initiative 2019." In: *Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC 2019)*. Vol. 2536. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 46–85. URL: http://ceur-ws.org/Vol-2536/oaei19_paper0.pdf.

[167] Mina Abd Nikooie Pour, Alsayed Algergawy, Reihaneh Amini, Daniel Faria, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Clement Jonquet, Naouel Karam, Abderrahmane Khiat, Amir Laadhar, Patrick Lambrix, Huanyu Li, Ying Li, Pascal Hitzler, Heiko Paulheim, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Élodie Thiéblin, Cássia Trojahn, Jana Vatascinová, Beyza Yaman, Ondrej Zamazal, and Lu Zhou. "Results of the Ontology Alignment Evaluation Initiative 2020." In: *Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020)*. Vol. 2788. CEUR Workshop Proceedings. CEUR-WS.org, pp. 92–138. URL: http://ceur-ws.org/Vol-2788/oaei20_paper0.pdf.

[168] Mina Abd Nikooie Pour, Alsayed Algergawy, Florence Amardeilh, Reihaneh Amini, Omaima Fallatah, Daniel Faria, Irini Fundulaki, Ian Harrow, Sven Hertling, Pascal Hitzler, Martin Huschka, Liliana Ibanescu, Ernesto Jiménez-Ruiz, Naouel Karam, Amir Laadhar, Patrick Lambrix, Huanyu Li, Ying Li, Franck Michel, Engy Nasr, Heiko Paulheim, Catia Pesquita, Jan Portisch, Catherine Roussey, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Cássia Trojahn, Jana Vatascinová,

Beyza Yaman, Ondrej Zamazal, and Lu Zhou. “Results of the Ontology Alignment Evaluation Initiative 2021.” In: *Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021)*. Vol. 3063. CEUR Workshop Proceedings. CEUR-WS.org, pp. 62–108. URL: `http://ceur-ws.org/Vol-3063/oaei21_paper0.pdf`.

[169] Ernesto Jiménez-Ruiz, Tzanina Saveta, Ondvrej Zamazal, Sven Hertling, Michael Röder, Irini Fundulaki, Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, Amina Annane, Zohra Bellahsene, Sadok Ben Yahia, Gayo Diallo, Daniel Faria, Marouen Kachroudi, Abderrahmane Khiat, Patrick Lambrix, Huanyu Li, Maximilian Mackeprang, Majid Mohammadi, Maciej Rybinski, Booma Sowkarthiga Balasubramani, and Cássia Trojahn. “Introducing the HOBBIT platform into the Ontology Alignment Evaluation Campaign.” In: *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference (ISWC 2018)*. Vol. 2288. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 49–60. URL: `http://ceur-ws.org/Vol-2288/om2018_LTpaper5.pdf`.

[170] Zlatan Dragisic, Valentina Ivanova, Patrick Lambrix, Daniel Faria, Ernesto Jiménez-Ruiz, and Catia Pesquita. “User Validation in Ontology Alignment.” In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016*. Vol. 9981. Lecture Notes in Computer Science. Springer, Cham, 2016, pp. 200–217. DOI: `10.1007/978-3-319-46523-4_13`.

[171] Huanyu Li, Zlatan Dragisic, Daniel Faria, Valentina Ivanova, Ernesto Jiménez-Ruiz, Patrick Lambrix, and Catia Pesquita. “User validation in ontology alignment: functional assessment and impact.” In: *The Knowledge Engineering Review* 34 (2019), e15. DOI: `10.1017/S0269888919000080`.

[172] Elodie Thiéblin, Michelle Cheatham, Cassia Trojahn, and Ondrej Zamazal. “A consensual dataset for complex ontology matching evaluation.” In: *The Knowledge Engineering Review* 35 (2020), e34. DOI: `10.1017/S0269888920000247`.

[173] Lu Zhou, Elodie Thiéblin, Michelle Cheatham, Daniel Faria, Catia Pesquita, Cassia Trojahn, and Ondřej Zamazal. “Towards evaluating

complex ontology alignments.” In: *The Knowledge Engineering Review* 35 (2020), e21. DOI: `10.1017/S0269888920000168`.

Appendix

A

SPARQL queries for MDO competency questions

This appendix lists the 14 SPARQL queries to answer competency questions covered in the requirements analysis of MDO presented in Chapter 5.

CQ1: What are the calculated properties and their values produced by a materials calculation?

Listing A.1: A SPARQL query for MDO CQ1.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>

SELECT ?calculation ?property ?value WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasOutputCalculatedProperty ?property.
  ?property core:hasPropertyValue ?value.
}
```

CQ2: What are the input and output structures of a materials calculation?

Listing A.2: A SPARQL query for MDO CQ2.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>

SELECT ?calculation ?input_structure ?output_structure WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasInputStructure ?input_structure;
               core:hasOutputStructure ?output_structure.
}
```

CQ3: What is the space group type of a structure?

Listing A.3: A SPARQL query for MDO CQ3.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX structure: <https://w3id.org/mdo/structure/>

SELECT ?calculation ?output_structure ?symbol WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasOutputStructure ?output_structure.
  ?output_structure rdf:type core:Structure;
                    structure:hasSpaceGroup ?spacegroup.
  ?spacegroup rdf:type structure:SpaceGroup;
              structure:hasSpaceGroupSymbol ?symbol.
}
```

CQ4: What is the lattice type of a structure?

Listing A.4: A SPARQL query for MDO CQ4.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX structure: <https://w3id.org/mdo/structure/>

SELECT ?calculation ?output_structure ?type WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasOutputStructure ?output_structure.
  ?output_structure rdf:type core:Structure;
                    structure:hasLattice ?lattice.
  ?lattice rdf:type structure:Lattice;
           structure:hasLatticeType ?type.
}
```

CQ5: What is the chemical formula of a structure?

Listing A.5: A SPARQL query for MDO CQ5.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX structure: <https://w3id.org/mdo/structure/>

SELECT ?calculation ?outputstructure ?formula WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasOutputStructure ?outputstructure.
  ?outputstructure structure:hasComposition ?composition.
  ?composition structure:hasDescriptiveFormula ?formula.
}
```

CQ6: For a series of materials calculations, what are the compositions of materials with a specific range of a calculated property (e.g., band gap)?

Listing A.6: A SPARQL query for MDO CQ6.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX structure: <https://w3id.org/mdo/structure/>
PREFIX qudt: <http://qudt.org/schema/qudt/>

SELECT ?formula ?value WHERE
{
  ?calculation rdf:type core:Calculation;
          core:hasOutputCalculatedProperty ?property;
          core:hasOutputStructure ?output_structure.
  ?property qudt:quantityValue ?quantity_value;
            core:hasPropertyName ?name.
  ?quantity_value rdf:type qudt:QuantityValue;
                  qudt:numericValue ?value.
  ?output_structure structure:hasComposition ?composition.
  ?composition structure:hasDescriptiveFormula ?formula.
  FILTER (?value>5 && ?name="band_gap")
}
```

CQ7: For a specific material and a given range of a calculated property (e.g., band gap), what is the lattice type of the structure?

Listing A.7: A SPARQL query for MDO CQ7.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX structure: <https://w3id.org/mdo/structure/>
PREFIX calculation: <https://w3id.org/mdo/calculation/>

SELECT ?outputstructure ?value ?type WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasOutputCalculatedProperty ?property;
               core:hasOutputStructure ?outputstructure.
  ?property core:hasPropertyValue ?value;
            core:hasPropertyName ?name.
  ?outputstructure structure:hasLattice ?lattice.
  ?lattice structure:hasLatticeType ?type.
  FILTER (?value>5 && ?name="band_gap")
}
```

CQ8: For a specific material and an expected lattice type of output structure, what are the values of calculated properties of the calculations?

Listing A.8: A SPARQL query for MDO CQ8.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX structure: <https://w3id.org/mdo/structure/>

SELECT ?outputstructure ?value ?type WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasOutputCalculatedProperty ?property;
               core:hasOutputStructure ?outputstructure.
  ?Property core:hasPropertyValue ?value;
            core:hasPropertyName ?name.
  ?outputstructure structure:hasLattice ?lattice.
  ?lattice structure:hasLatticeType ?type.
  FILTER (?name="band_gap" && ?type="cubic")
}
```

CQ9: What is the computational method used in a materials calculation?

Listing A.9: A SPARQL query for MDO CQ9.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX calculation: <https://w3id.org/mdo/calculation/>

SELECT ?calculation ?method WHERE
{
  ?calculation rdf:type core:Calculation;
               calculation:hascomputationalMethod ?method.
}
```

CQ10: What is the value for a specific parameter (e.g., cutoff energy) of the method used for the calculation?

Listing A.10: A SPARQL query for MDO CQ10.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX calculation: <https://w3id.org/mdo/calculation/>

SELECT ?calculation ?method ?name ?value WHERE
{
  ?calculation rdf:type core:Calculation;
               calculation:hasComputationalMethod ?method.
  ?method calculation:hasParameter ?parameter;
             calculation:hasParameterValue ?value;
             calculation:hasParameterName ?name.
  FILTER (?name="cutoff_energy")
}
```

CQ11: Which software produced the result of a calculation?

Listing A.11: A SPARQL query for MDO CQ11.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX prov: <http://www.w3.org/ns/prov#>

SELECT ?calculation ?software WHERE
{
  ?calculation rdf:type core:Calculation;
               prov:wasAssociatedWith ?software.
}
```

CQ12: Who are the authors of the calculation?

Listing A.12: A SPARQL query for MDO CQ12.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX provenance: <https://w3id.org/mdo/provenance/>
PREFIX prov: <http://www.w3.org/ns/prov#>

SELECT ?calculation ?author_name WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasOutputStructure ?output_structure.
  ?output_structure rdf:type core:Structure;
                    prov:wasAttributedTo ?reference.
  ?reference rdf:type provenance:ReferenceAgent;
             provenance:hasAuthorName ?author_name.
}
```

CQ13: Which software or code does the calculation run with?

Listing A.13: A SPARQL query for MDO CQ13.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX prov: <http://www.w3.org/ns/prov#>

SELECT ?calculation ?software WHERE
{
  ?calculation rdf:type core:Calculation;
               prov:wasAssociatedWith ?software.
}
```

CQ14: When was the calculation data published to the database?

Listing A.14: A SPARQL query for MDO CQ14.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <https://w3id.org/mdo/core/>
PREFIX provenance: <https://w3id.org/mdo/provenance/>
PREFIX prov: <http://www.w3.org/ns/prov#>

SELECT ?calculation ?date WHERE
{
  ?calculation rdf:type core:Calculation;
               core:hasOutputStructure ?output_structure.
  ?output_structure rdf:type core:Structure;
                    prov:wasAttributedTo ?reference.
  ?reference rdf:type provenance:ReferenceAgent;
             provenance:hasPublicationDateTime ?datetime.
}
```

Appendix

B GraphQL schemas used in the evaluation

This appendix lists the GraphQL schemas used in the real case evaluation and the evaluation based on LinGBM, presented in Chapter 8.

B.1 MDO related GraphQL schema

Listing B.1: MDO related GraphQL schema.

```
interface Thing{
  iri: String
}
interface Property{
  PropertyName: String
  numericalValue: Float
  iri: String
}
type Query{
  PhysicalPropertyList(filter: PhysicalPropertyFilter):
       [PhysicalProperty]
  AngleTripleList(filter: AngleTripleFilter): [AngleTriple]
  CompositionList(filter: CompositionFilter): [Composition]
  CalculatedPropertyList(filter: CalculatedPropertyFilter):
       [CalculatedProperty]
  AxisVectorsList(filter: AxisVectorsFilter): [AxisVectors]
  LatticeList(filter: LatticeFilter): [Lattice]
  OccupancyList(filter: OccupancyFilter): [Occupancy]
```

```
    SpeciesList(filter: SpeciesFilter): [Species]
    BasisList(filter: BasisFilter): [Basis]
    LengthTripleList(filter: LengthTripleFilter): [LengthTriple]
    SpaceGroupList(filter: SpaceGroupFilter): [SpaceGroup]
    StructureList(filter: StructureFilter): [Structure]
    CalculationList(filter: CalculationFilter): [Calculation]
    CoordinateVectorList(filter: CoordinateVectorFilter):
         [CoordinateVector]
    PointGroupList(filter: PointGroupFilter): [PointGroup]
    SiteList(filter: SiteFilter): [Site]
    PropertyList(filter: PropertyFilter): [Property]
  }
  type AxisVectors{
    has_c_axisVector: CoordinateVector
    has_b_axisVector: CoordinateVector
    has_a_axisVector: CoordinateVector
    iri: String
  }
  type Lattice{
    hasAngleVector: AngleTriple
    hasLengthVector: LengthTriple
    hasAxisVectors: AxisVectors
    iri: String
  }
  type CoordinateVector{
    X_axisCoordinate: Float
    Z_axisCoordinate: Float
    Y_axisCoordinate: Float
    iri: String
  }
  type CalculatedProperty implements Property{
    PropertyName: String
    numericalValue: Float
    iri: String
  }
  type PhysicalProperty implements Property{
    PropertyName: String
    numericalValue: Float
    iri: String
  }
```

```
type Composition{
  ReducedFormula: String
  HillFormula: String
  DescriptiveFormula: String
  AnonymousFormula: String
  iri: String
}
type Occupancy{
  hasSpecies: [Species]
  hasSite: [Site]
  iri: String
}
type Structure implements  Thing{
  hasOccupancy: [Occupancy]
  hasSpaceGroup: SpaceGroup
  hasComposition: Composition
  hasBasis: Basis
  hasLattice: Lattice
  iri: String
}
type Calculation implements  Thing{
  ID: String
  hasInputProperty: Property
  hasOutputCalculatedProperty: CalculatedProperty
  hasInputStructure: [Structure]
  hasOutputStructure: [Structure]
  iri: String
}
type PointGroup{
  PointGroupHMName: String
  iri: String
}
type SpaceGroup{
  hasPointGroup: PointGroup
  SpaceGroupID: Int
  SpaceGroupSymbol: String
  iri: String
}
type LengthTriple{
  Length_a: Float
```

```
    Length_b: Float
    Length_c: Float
    iri: String
  }
  type Site{
    hasCartesianCoordinates: CoordinateVector
    hasFractionalCoordinates: CoordinateVector
    iri: String
  }
  type AngleTriple{
    Angle_gamma: Float
    Angle_alpha: Float
    Angle_beta: Float
    iri: String
  }
  type Basis{
    hasAxisVectors: [AxisVectors]
    hasAngleVector: [AngleTriple]
    hasLengthVector: [LengthTriple]
    iri: String
  }
  type Species{
    iri: String
  }
  input AxisVectorsFilter{
    _and: [AxisVectorsFilter]
    _or: [AxisVectorsFilter]
    _not: AxisVectorsFilter
    has_c_axisVector: CoordinateVectorFilter
    has_b_axisVector: CoordinateVectorFilter
    has_a_axisVector: CoordinateVectorFilter
    iri: StringFilter
  }
  input LatticeFilter{
    _and: [LatticeFilter]
    _or: [LatticeFilter]
    _not: LatticeFilter
    hasAngleVector: AngleTripleFilter
    hasLengthVector: LengthTripleFilter
    hasAxisVectors: AxisVectorsFilter
    iri: StringFilter
  }
  input CoordinateVectorFilter{
    _and: [CoordinateVectorFilter]
    _or: [CoordinateVectorFilter]
    _not: CoordinateVectorFilter
    X_axisCoordinate: FloatFilter
    Z_axisCoordinate: FloatFilter
    Y_axisCoordinate: FloatFilter
    iri: StringFilter
  }
  input PropertyFilter{
    _and: [PropertyFilter]
    _or: [PropertyFilter]
    _not: PropertyFilter
    numericalValue: FloatFilter
    iri: StringFilter
  }
  input CalculatedPropertyFilter{
    _and: [CalculatedPropertyFilter]
    _or: [CalculatedPropertyFilter]
    _not: CalculatedPropertyFilter
    numericalValue: FloatFilter
    iri: StringFilter
  }
  input PhysicalPropertyFilter{
    _and: [PhysicalPropertyFilter]
    _or: [PhysicalPropertyFilter]
    _not: PhysicalPropertyFilter
    numericalValue: FloatFilter
    iri: StringFilter
  }
  input CompositionFilter{
    _and: [CompositionFilter]
    _or: [CompositionFilter]
    _not: CompositionFilter
    ReducedFormula: StringFilter
    HillFormula: StringFilter
    DescriptiveFormula: StringFilter
    AnonymousFormula: StringFilter
    iri: StringFilter
  }
  input OccupancyFilter{
    _and: [OccupancyFilter]
    _or: [OccupancyFilter]
    _not: OccupancyFilter
    hasSpecies: SpeciesFilter
    hasSite: SiteFilter
    iri: StringFilter
  }
  input StructureFilter{
    _and: [StructureFilter]
    _or: [StructureFilter]
    _not: StructureFilter
    hasOccupancy: OccupancyFilter
    hasSpaceGroup: SpaceGroupFilter
    hasComposition: CompositionFilter
    hasBasis: BasisFilter
    hasLattice: LatticeFilter
    iri: StringFilter
  }
  input CalculationFilter{
    _and: [CalculationFilter]
    _or: [CalculationFilter]
    _not: CalculationFilter
    ID: StringFilter
    hasInputProperty: PropertyFilter
    hasOutputCalculatedProperty: CalculatedPropertyFilter
    hasInputStructure: StructureFilter
    hasOutputStructure: StructureFilter
    iri: StringFilter
  }
  input PointGroupFilter{
    _and: [PointGroupFilter]
    _or: [PointGroupFilter]
    _not: PointGroupFilter
    PointGroupHMName: StringFilter
    iri: StringFilter
  }
  input SpaceGroupFilter{
    _and: [SpaceGroupFilter]
    _or: [SpaceGroupFilter]
    _not: SpaceGroupFilter
    hasPointGroup: PointGroupFilter
    SpaceGroupID: IntFilter
    SpaceGroupSymbol: StringFilter
    iri: StringFilter
  }
  input LengthTripleFilter{
    _and: [LengthTripleFilter]
    _or: [LengthTripleFilter]
    _not: LengthTripleFilter
    Length_a: FloatFilter
    Length_b: FloatFilter
    Length_c: FloatFilter
    iri: StringFilter
  }
  input SiteFilter{
    _and: [SiteFilter]
    _or: [SiteFilter]
    _not: SiteFilter
    hasCartesianCoordinates: CoordinateVectorFilter
    hasFractionalCoordinates: CoordinateVectorFilter
    iri: StringFilter
  }
  input AngleTripleFilter{
    _and: [AngleTripleFilter]
    _or: [AngleTripleFilter]
    _not: AngleTripleFilter
    Angle_gamma: FloatFilter
    Angle_alpha: FloatFilter
    Angle_beta: FloatFilter
    iri: StringFilter
  }
  input BasisFilter{
    _and: [BasisFilter]
    _or: [BasisFilter]
    _not: BasisFilter
    hasAxisVectors: AxisVectorsFilter
    hasAngleVector: AngleTripleFilter
    hasLengthVector: LengthTripleFilter
    iri: StringFilter
  }
  input SpeciesFilter{
    _and: [SpeciesFilter]
    _or: [SpeciesFilter]
    _not: SpeciesFilter
    iri: StringFilter
  }
  input StringFilter{
    _eq: String
    _neq: String
    _gt: String
    _egt: String
    _lt: String
    _elt: String
    _in: [String]
    _nin: [String]
    _like: String
    _ilike: String
  }
  input IntFilter{
    _eq: Int
    _neq: Int
    _gt: Int
    _egt: Int
    _lt: Int
    _elt: Int
    _in: [Int]
    _nin: [Int]
    _like: Int
    _ilike: Int
  }
  input FloatFilter{
    _eq: Float
    _neq: Float
    _gt: Float
    _egt: Float
    _lt: Float
    _elt: Float
    _in: [Float]
    _nin: [Float]
    _like: Float
    _ilike: Float
  }
```

B.2 University related GraphQL schema

Listing B.2: University related GraphQL schema.

```
type Query{
  UniversityList(filter: UniversityFilter): [University]
  FacultyList(filter: FacultyFilter): [Faculty]
  DepartmentList(filter: DepartmentFilter): [Department]
  ResearchGroupList(filter: ResearchGroupFilter): [ResearchGroup]
  ProfessorList(filter: ProfessorFilter): [Professor]
  LecturerList(filter: LecturerFilter): [Lecturer]
  PublicationList(filter: PublicationFilter): [Publication]
  GraduateStudentList(filter: GraduateStudentFilter):
        [GraduateStudent]
}
type University{
  nr: Int
  name: String
  undergraduateDegreeObtainedByFaculty: [Faculty]
  mastergraduateDegreeObtainers: [Faculty]
  doctoralDegreeObtainers: [Faculty]
  undergraduateDegreeObtainedBystudent: [GraduateStudent]
}
type Faculty{
  nr: Int
  name: String
  telephone: String
  emailAddress: String
  undergraduateDegreeFrom: University
  masterDegreeFrom: University
  doctoralDegreeFrom: University
  worksFor: Department
  publications: [Publication]
}
type Department{
  nr: Int
  name: String
  subOrganizationOf: University
  head: Professor
  faculties: [Faculty]
```

```
}
type ResearchGroup{
  nr: Int
  subOrganizationOf: Department
}
type Professor{
  nr: Int
  professorType: String
  researchInterest: String
  headOf: Department
  name: String
  telephone: String
  emailAddress: String
  undergraduateDegreeFrom: University
  masterDegreeFrom: University
  doctoralDegreeFrom: University
  worksFor: Department
  publications: [Publication]
}
type Lecturer{
  nr: Int
  name: String
  telephone: String
  emailAddress: String
  undergraduateDegreeFrom: University
  masterDegreeFrom: University
  doctoralDegreeFrom: University
  worksFor: Department
  publications: [Publication]
}
type Publication{
  nr: Int
  name: String
  title: String
  abstract: String
  mainAuthor: [Faculty]
}
type GraduateStudent{
  nr: Int
  name: String
```

```
    telephone: String
    emailAddress: String
    age: Int
    memberOf: Department
    undergraduateDegreeFrom: University
    advisor: Professor
}
input UniversityFilter{
    _and: [UniversityFilter]
    _or: [UniversityFilter]
    _not: UniversityFilter
    nr: IntFilter
    name: StringFilter
    undergraduateDegreeObtainedByFaculty: [FacultyFilter]
    mastergraduateDegreeObtainers: [FacultyFilter]
    doctoralDegreeObtainers: [FacultyFilter]
    undergraduateDegreeObtainedBystudent: [GraduateStudentFilter]
}
input FacultyFilter{
    _and: [FacultyFilter]
    _or: [FacultyFilter]
    _not: FacultyFilter
    nr: IntFilter
    name: StringFilter
    telephone: StringFilter
    emailAddress: StringFilter
    undergraduateDegreeFrom: UniversityFilter
    masterDegreeFrom: UniversityFilter
    doctoralDegreeFrom: UniversityFilter
    worksFor: DepartmentFilter
    publications: [PublicationFilter]
}
input DepartmentFilter{
    _and: [DepartmentFilter]
    _or: [DepartmentFilter]
    _not: DepartmentFilter
    nr: IntFilter
    name: StringFilter
    subOrganizationOf: UniversityFilter
    head: ProfessorFilter
```

```
  faculties: [FacultyFilter]
}
input ResearchGroupFilter{
  _and: [ResearchGroupFilter]
  _or: [ResearchGroupFilter]
  _not: ResearchGroupFilter
  nr: IntFilter
  subOrganizationOf: DepartmentFilter
}
input ProfessorFilter{
  _and: [ProfessorFilter]
  _or: [ProfessorFilter]
  _not: ProfessorFilter
  nr: IntFilter
  professorType: StringFilter
  researchInterest: StringFilter
  headOf: StringFilter
  name: StringFilter
  telephone: StringFilter
  emailAddress: StringFilter
  undergraduateDegreeFrom: UniversityFilter
  masterDegreeFrom: UniversityFilter
  doctoralDegreeFrom: UniversityFilter
  worksFor: DepartmentFilter
  publications: [PublicationFilter]
}
input LecturerFilter{
  _and: [LecturerFilter]
  _or: [LecturerFilter]
  _not: LecturerFilter
  nr: IntFilter
  name: StringFilter
  telephone: StringFilter
  emailAddress: StringFilter
  undergraduateDegreeFrom: UniversityFilter
  masterDegreeFrom: UniversityFilter
  doctoralDegreeFrom: UniversityFilter
  worksFor: DepartmentFilter
  publications: [PublicationFilter]
}
```

```
input PublicationFilter{
   _and: [PublicationFilter]
   _or: [PublicationFilter]
   _not: PublicationFilter
   nr: IntFilter
   name: StringFilter
   title: StringFilter
   abstract: StringFilter
   mainAuthor: [FacultyFilter]
}
input GraduateStudentFilter{
   _and: [GraduateStudentFilter]
   _or: [GraduateStudentFilter]
   _not: GraduateStudentFilter
   nr: IntFilter
   name: StringFilter
   telephone: StringFilter
   emailAddress: StringFilter
   age: IntFilter
   memberOf: DepartmentFilter
   undergraduateDegreeFrom: UniversityFilter
   advisor: ProfessorFilter
}
input StringFilter{
   _eq: String
   _neq: String
   _gt: String
   _egt: String
   _lt: String
   _elt: String
   _in: [String]
   _nin: [String]
   _like: String
   _ilike: String
}
input IntFilter{
   _eq: Int
   _neq: Int
   _gt: Int
   _egt: Int
```

```
    _lt: Int
    _elt: Int
    _in: [Int]
    _nin: [Int]
    _like: Int
    _ilike: Int
  }
```

APPENDIX C

GraphQL queries used in the evaluation

This appendix lists the 12 GraphQL queries used in the real case evaluation and 8 example queries used in the evaluation based on LinGBM, presented in Chapter 8.

C.1 MDO related queries

C.1.1 Queries without filter expressions

Query 1: List all the structures containing the reduced formula of each structure's composition.

Listing C.1: Q1 in the real case evaluation.

```
{
  StructureList{
    hasComposition{
      ReducedFormula
    }
  }
}
```

Query 2: List all the calculations containing the reduced formula of each output structure's composition.

Listing C.2: Q2 in the real case evaluation.

```
{
  CalculationList{
    hasOutputStructure{
      hasComposition{
        ReducedFormula
      }
    }
  }
}
```

Query 3: List all the calculations containing the name and value of each output calculated property.

Listing C.3: Q3 in the real case evaluation.

```
{
  CalculationList{
    hasOutputCalculatedProperty{
      PropertyName
      numericalValue
    }
  }
}
```

Query 4: List all the calculations containing the name and value of each output calculated property, the reduced formula of each output structure's composition.

Listing C.4: Q4 in the real case evaluation.

```
{
  CalculationList{
    hasOutputStructure{
      hasComposition{
        ReducedFormula
      }
    }
    hasOutputCalculatedProperty{
      PropertyName
      numericalValue
    }
  }
}
```

Query 5: List all the calculations and structures.

Listing C.5: Q5 in the real case evaluation.

```
{
  ThingList{
    ... on Calculation{iri}
    ... on Structure{iri}
  }
}
```

C.1.2 Queries with filter expressions

Query 6: List all the calculations where the ID is in a given list of values.

Listing C.6: Q6 in the real case evaluation.

```
{
  CalculationList(
    filter: {
      ID: {
        _in: [ "6332","8088","21331","mp-561628","mp-614918" ]
      }
    }
  )
  {
    ID
    hasOutputCalculatedProperty {
      PropertyName
      numericalValue
    }
  }
}
```

Query 7: List all the calculations where the ID is in a given list of values and the reduced formula is in a given list of values.

Listing C.7: Q7 in the real case evaluation.

```
{
  CalculationList(
    filter: {
      _and: [
        {
          ID: {
           _in: [ "6332","8088","21331","mp-561628","mp-614918" ]
          }
        }
        {
          hasOutputStructure: {
            hasComposition: {
              ReducedFormula: {
                _in: [ "MnCl2","YClO" ]
              }
            }
          }
        }
      ]
    }
  )
  {
    ID
    hasOutputCalculatedProperty {
      PropertyName
      numericalValue
    }
  }
}
```

Query 8: List all the calculations where the ID is in a given list of values, and the reduced formula is in a given list A or B.

Listing C.8: Q8 in the real case evaluation.

```
{
  CalculationList(
    filter: {
      _and: [
        {
          ID: {
            _in: ["6332","8088","21331","mp-561628","mp-614918"]}
        }
        {
          _or: [
            {
              hasOutputStructure: { hasComposition: {
                  ReducedFormula: { _in: [ "MnCl2","YClO" ]}
                }
              }
            }
            {
              hasOutputStructure: { hasComposition: {
                  ReducedFormula: { _in: ["CeCrS2O","SiO2","O"]}
                }
              }
            }
          ]
        }
      ]
    }
  )
  {
    ID
    hasOutputCalculatedProperty {
      PropertyName
      numericalValue
    }
  }
}
```

Query 9: List all the calculations where the value of band gap property is higher than 5.

Listing C.9: Q9 in the real case evaluation.

```
{
  CalculationList(
    filter: {
      hasOutputCalculatedProperty: {
        _and: [
          { PropertyName: { _eq: "Band Gap" } }
          { numericalValue: { _gt: 5 } }
        ]
      }
    }
  )
  {
    ID
    hasOutputStructure {
      hasComposition {
        ReducedFormula
      }
    }
  }
}
```

Query 10: List all the calculations where the value of band gap property is higher than 5, and the reduced formula in a given list of values.

Listing C.10: Q10 in the real case evaluation.

```
{
  CalculationList(
    filter: {
      _and: [
        {
          hasOutputStructure: {
            hasComposition: {
              ReducedFormula: { _in: [ "MnCl2", "YClO" ] }
            }
          }
        }
        {
          hasOutputCalculatedProperty: {
            _and: [
              { PropertyName: { _eq: "Band Gap" } }
              { numericalValue: { _gt: 5 } }
            ]
          }
        }
      ]
    }
  )
  {
    ID
    hasOutputStructure {
      hasComposition {
        ReducedFormula
      }
    }
    hasOutputCalculatedProperty {
      PropertyName
      numericalValue
    }
  }
}
```

Query 11: List all the calculations where the filter condition is complex that needs to be simplified.

Listing C.11: Q11 in the real case evaluation.

```
{
  CalculationList(
    filter: {
      _and: [
        { hasOutputCalculatedProperty: {
            _and: [
              { PropertyName: { _eq: "Band Gap" } }
              { numericalValue: { _gt: 4 } }
            ]
          }
        }
        {
          _or: [
            { hasOutputCalculatedProperty: {
                _and: [
                  { PropertyName: { _eq: "Band Gap" } }
                  { numericalValue: { _gt: 4 } }
                ]
              }
            }
            { hasOutputStructure: {
                hasComposition: {
                  ReducedFormula: { _in: [ "YClO", "CsCl" ] }
                }
              }
            }
          ]
        }
      ]
    }
  )
  {
    ID
  }
}
```

Query 12: List all the structures that contain Silicon element.

Listing C.12: Q12 in the real case evaluation.

```
{
  StructureList(
    filter: {
      hasComposition: {
        ReducedFormula: { _like: "%Si%" }
      }
    }
  )
  {
    hasComposition {
      ReducedFormula
    }
  }
}
```

C.2 Query examples according to query templates in LinGBM.

An example query in QS1 according to QT1. Queries of this template retrieve several attributes of the graduate student that get bachelor's degree from the university that grant the doctoral degree to the given faculty.

Listing C.13: An example query based on QT1 from LinGBM.

```
{
  FacultyList(
    filter: {
      nr: { _eq: 214041 }
    }
  )
  {
    doctoralDegreeFrom {
      undergraduateDegreeObtainedBystudent {
        nr
        emailAddress
      }
    }
  }
}
```

An example query in QS2 according to QT2. Queries of this template retrieve all the publications by all faculties that got their doctoral degree from a given university.

Listing C.14: An example query based on QT2 from LinGBM.

```
{
  UniversityList(
    filter: {
      nr: { _eq: 531 }
    }
  )
  {
    doctoralDegreeObtainers {
      publications {
        title
      }
    }
  }
}
```

An example query in QS3 according to QT3. Given a research group that belongs to a department, queries of this template retrieve the University that granted the doctoral degree to the head of this department.

Listing C.15: An example query based on QT3 from LinGBM.

```
{
  ResearchGroupList(
    filter: {
      nr: { _eq: 32008 }
    }
  )
  {
    subOrganizationOf {
      head {
        nr
        emailAddress
        doctoralDegreeFrom {
          nr
        }
      }
    }
  }
}
```

An example query in QS4 according to QT4. Queries of this template retrieve the details of the graduate student that got bachelor' s degree from the same university as the one that granted the doctoral degree to the given lecturer, including the department of the students' supervisor.

Listing C.16: An example query based on QT4 from LinGBM.

```
{
  LecturerList(
    filter: {
      nr: { _eq: 209064 }
    }
  )
  {
    doctoralDegreeFrom {
      nr
      undergraduateDegreeObtainedBystudent {
        nr
        emailAddress
        advisor {
          nr
          emailAddress
          worksFor {
            nr
          }
        }
      }
    }
  }
}
```

An example query in QS5 according to QT5. Queries of this template go from a given department to its university, then retrieve all graduate students who got the bachelor's degree from the university, then come back to the department. Each query repeats this cycle two times and requests the students' email addresses along the way.

Listing C.17: An example query based on QT5 from LinGBM.

```
{
  DepartmentList(
    filter:{
      nr:{ _eq: 314 }
    })
  {
    nr
    subOrganizationOf {
      nr
      undergraduateDegreeObtainedBystudent {
        nr
        emailAddress
        memberOf {
          nr
          subOrganizationOf {
            nr
            undergraduateDegreeObtainedBystudent {
              nr
              emailAddress
              memberOf {
                nr
              }
            }
          }
        }
      }
    }
  }
}
```

An example query in QS6 according to QT6. Queries of this template retrieve all graduate students that graduated from a given university, and then retrieve the professors that supervise these students and the department' s head of these professors.

Listing C.18: An example query based on QT6 from LinGBM.

```
{
  UniversityList(
    filter: {
      nr: { _eq: 973 }
    }
  )
  {
    undergraduateDegreeObtainedBystudent {
      advisor {
        worksFor {
          nr
        }
    }
    }
  }
}
```

An example query in QS7 according to QT10. Queries of this template retrieve all publications for which the title contains the given keyword.

Listing C.19: An example query based on QT10 from LinGBM.

```
{
  PublicationList(
    filter: {
      title:{ _like: "%potsy%" }
    }
  )
  {
    nr
    title
    abstract
  }
}
```

An example query in QS8 according to QT11. Queries of this template search for all graduate students who have graduated from a given university by using a search condition (instead of starting the traversal from the given university as done in Q6). Then, for each graduate student, the advisor is requested.

Listing C.20: An example query based on QT11 from LinGBM.

```
{
  GraduateStudentList(
    filter: {
      undergraduateDegreeFrom: {
        nr: { _eq: 424 }
      }
    }
  )
  {
    nr
    advisor {
      nr
    }
  }
}
```

No 918 **Jonas Lundberg:** Shaping Electronic News: Genre Perspectives on Interaction Design, 2004, ISBN 91-85297-14-3.

No 900 **Mattias Arvola:** Shades of use: The dynamics of interaction design for sociable use, 2004, ISBN 91-85295-42-6.

No 920 **Luis Alejandro Cortés:** Verification and Scheduling Techniques for Real-Time Embedded Systems, 2004, ISBN 91-85297-21-6.

No 929 **Diana Szentivanyi:** Performance Studies of Fault-Tolerant Middleware, 2005, ISBN 91-85297-58-5.

No 933 **Mikael Cäker:** Management Accounting as Constructing and Opposing Customer Focus: Three Case Studies on Management Accounting and Customer Relations, 2005, ISBN 91-85297-64-X.

No 937 **Jonas Kvarnström:** TALplanner and Other Extensions to Temporal Action Logic, 2005, ISBN 91-85297-75-5.

No 938 **Bourhane Kadmiry:** Fuzzy Gain-Scheduled Visual Servoing for Unmanned Helicopter, 2005, ISBN 91-85297-76-3.

No 945 **Gert Jervan:** Hybrid Built-In Self-Test and Test Generation Techniques for Digital Systems, 2005, ISBN 91-85297-97-6.

No 946 **Anders Arpteg:** Intelligent Semi-Structured Information Extraction, 2005, ISBN 91-85297-98-4.

No 947 **Ola Angelsmark:** Constructing Algorithms for Constraint Satisfaction and Related Problems - Methods and Applications, 2005, ISBN 91-85297-99-2.

No 963 **Calin Curescu:** Utility-based Optimisation of Resource Allocation for Wireless Networks, 2005, ISBN 91-85457-07-8.

No 972 **Björn Johansson:** Joint Control in Dynamic Situations, 2005, ISBN 91-85457-31-0.

No 974 **Dan Lawesson:** An Approach to Diagnosability Analysis for Interacting Finite State Systems, 2005, ISBN 91-85457-39-6.

No 979 **Claudiu Duma:** Security and Trust Mechanisms for Groups in Distributed Services, 2005, ISBN 91-85457-54-X.

No 983 **Sorin Manolache:** Analysis and Optimisation of Real-Time Systems with Stochastic Behaviour, 2005, ISBN 91-85457-60-4.

No 986 **Yuxiao Zhao:** Standards-Based Application Integration for Business-to-Business Communications, 2005, ISBN 91-85457-66-3.

No 1004 **Patrik Haslum:** Admissible Heuristics for Automated Planning, 2006, ISBN 91-85497-28-2.

No 1005 **Aleksandra Tešanovic:** Developing Reusable and Reconfigurable Real-Time Software using Aspects and Components, 2006, ISBN 91-85497-29-0.

No 1008 **David Dinka:** Role, Identity and Work: Extending the design and development agenda, 2006, ISBN 91-85497-42-8.

No 1009 **Iakov Nakhimovski:** Contributions to the Modeling and Simulation of Mechanical Systems with Detailed Contact Analysis, 2006, ISBN 91-85497-43-X.

No 1013 **Wilhelm Dahllöf:** Exact Algorithms for Exact Satisfiability Problems, 2006, ISBN 91-85523-97-6.

No 1016 **Levon Saldamli:** PDEModelica - A High-Level Language for Modeling with Partial Differential Equations, 2006, ISBN 91-85523-84-4.

No 1017 **Daniel Karlsson:** Verification of Component-based Embedded System Designs, 2006, ISBN 91-85523-79-8

No 1018 **Ioan Chisalita:** Communication and Networking Techniques for Traffic Safety Systems, 2006, ISBN 91-85523-77-1.

No 1019 **Tarja Susi:** The Puzzle of Social Activity - The Significance of Tools in Cognition and Cooperation, 2006, ISBN 91-85523-71-2.

No 1021 **Andrzej Bednarski:** Integrated Optimal Code Generation for Digital Signal Processors, 2006, ISBN 91-85523-69-0.

No 1022 **Peter Aronsson:** Automatic Parallelization of Equation-Based Simulation Programs, 2006, ISBN 91-85523-68-2.

No 1030 **Robert Nilsson:** A Mutation-based Framework for Automated Testing of Timeliness, 2006, ISBN 91-85523-35-6.

No 1034 **Jon Edvardsson:** Techniques for Automatic Generation of Tests from Programs and Specifications, 2006, ISBN 91-85523-31-3.

No 1035 **Vaida Jakoniene:** Integration of Biological Data, 2006, ISBN 91-85523-28-3.

No 1045 **Genevieve Gorrell:** Generalized Hebbian Algorithms for Dimensionality Reduction in Natural Language Processing, 2006, ISBN 91-85643-88-2.

No 1051 **Yu-Hsing Huang:** Having a New Pair of Glasses - Applying Systemic Accident Models on Road Safety, 2006, ISBN 91-85643-64-5.

No 1054 **Åsa Hedenskog:** Perceive those things which cannot be seen - A Cognitive Systems Engineering perspective on requirements management, 2006, ISBN 91-85643-57-2.

No 1061 **Cécile Åberg:** An Evaluation Platform for Semantic Web Technology, 2007, ISBN 91-85643-31-9.

No 1073 **Mats Grindal:** Handling Combinatorial Explosion in Software Testing, 2007, ISBN 978-91-85715-74-9.

No 1075 **Almut Herzog:** Usable Security Policies for Runtime Environments, 2007, ISBN 978-91-85715-65-7.

No 1079 **Magnus Wahlström:** Algorithms, measures, and upper bounds for Satisfiability and related problems, 2007, ISBN 978-91-85715-55-8.

No 1083 **Jesper Andersson:** Dynamic Software Architectures, 2007, ISBN 978-91-85715-46-6.

No 1086 **Ulf Johansson:** Obtaining Accurate and Comprehensible Data Mining Models - An Evolutionary Approach, 2007, ISBN 978-91-85715-34-3.

No 1089 **Traian Pop:** Analysis and Optimisation of Distributed Embedded Systems with Heterogeneous Scheduling Policies, 2007, ISBN 978-91-85715-27-5.

No 1091 **Gustav Nordh:** Complexity Dichotomies for CSP-related Problems, 2007, ISBN 978-91-85715-20-6.

No 1106 **Per Ola Kristensson:** Discrete and Continuous Shape Writing for Text Entry and Control, 2007, ISBN 978-91-85831-77-7.

No 1110 **He Tan:** Aligning Biomedical Ontologies, 2007, ISBN 978-91-85831-56-2.

No 1112 **Jessica Lindblom:** Minding the body - Interacting socially through embodied action, 2007, ISBN 978-91-85831-48-7.

No 1113 **Pontus Wärnestål:** Dialogue Behavior Management in Conversational Recommender Systems, 2007, ISBN 978-91-85831-47-0.

No 1120 **Thomas Gustafsson:** Management of Real-Time Data Consistency and Transient Overloads in Embedded Systems, 2007, ISBN 978-91-85831-33-3.

No 1127 **Alexandru Andrei:** Energy Efficient and Predictable Design of Real-time Embedded Systems, 2007, ISBN 978-91-85831-06-7.

No 1139 **Per Wikberg:** Eliciting Knowledge from Experts in Modeling of Complex Systems: Managing Variation and Interactions, 2007, ISBN 978-91-85895-66-3.

No 1143 **Mehdi Amirijoo:** QoS Control of Real-Time Data Services under Uncertain Workload, 2007, ISBN 978-91-85895-49-6.

No 1150 **Sanny Syberfeldt:** Optimistic Replication with Forward Conflict Resolution in Distributed Real-Time Databases, 2007, ISBN 978-91-85895-27-4.

No 1155 **Beatrice Alenljung:** Envisioning a Future Decision Support System for Requirements Engineering - A Holistic and Human-centred Perspective, 2008, ISBN 978-91-85895-11-3.

No 1156 **Artur Wilk:** Types for XML with Application to Xcerpt, 2008, ISBN 978-91-85895-08-3.

No 1183 **Adrian Pop:** Integrated Model-Driven Development Environments for Equation-Based Object-Oriented Languages, 2008, ISBN 978-91-7393-895-2.

No 1185 **Jörgen Skågeby:** Gifting Technologies - Ethnographic Studies of End-users and Social Media Sharing, 2008, ISBN 978-91-7393-892-1.

No 1187 **Imad-Eldin Ali Abugessaisa:** Analytical tools and information-sharing methods supporting road safety organizations, 2008, ISBN 978-91-7393-887-7.

No 1204 **H. Joe Steinhauer:** A Representation Scheme for Description and Reconstruction of Object Configurations Based on Qualitative Relations, 2008, ISBN 978-91-7393-823-5.

No 1222 **Anders Larsson:** Test Optimization for Core-based System-on-Chip, 2008, ISBN 978-91-7393-768-9.

No 1238 **Andreas Borg:** Processes and Models for Capacity Requirements in Telecommunication Systems, 2009, ISBN 978-91-7393-700-9.

No 1240 **Fredrik Heintz:** DyKnow: A Stream-Based Knowledge Processing Middleware Framework, 2009, ISBN 978-91-7393-696-5.

No 1241 **Birgitta Lindström:** Testability of Dynamic Real-Time Systems, 2009, ISBN 978-91-7393-695-8.

No 1244 **Eva Blomqvist:** Semi-automatic Ontology Construction based on Patterns, 2009, ISBN 978-91-7393-683-5.

No 1249 **Rogier Woltjer:** Functional Modeling of Constraint Management in Aviation Safety and Command and Control, 2009, ISBN 978-91-7393-659-0.

No 1260 **Gianpaolo Conte:** Vision-Based Localization and Guidance for Unmanned Aerial Vehicles, 2009, ISBN 978-91-7393-603-3.

No 1262 **AnnMarie Ericsson:** Enabling Tool Support for Formal Analysis of ECA Rules, 2009, ISBN 978-91-7393-598-2.

No 1266 **Jiri Trnka:** Exploring Tactical Command and Control: A Role-Playing Simulation Approach, 2009, ISBN 978-91-7393-571-5.

No 1268 **Bahlol Rahimi:** Supporting Collaborative Work through ICT - How End-users Think of and Adopt Integrated Health Information Systems, 2009, ISBN 978-91-7393-550-0.

No 1274 **Fredrik Kuivinen:** Algorithms and Hardness Results for Some Valued CSPs, 2009, ISBN 978-91-7393-525-8.

No 1281 **Gunnar Mathiason:** Virtual Full Replication for Scalable Distributed Real-Time Databases, 2009, ISBN 978-91-7393-503-6.

No 1290 **Viacheslav Izosimov:** Scheduling and Optimization of Fault-Tolerant Distributed Embedded Systems, 2009, ISBN 978-91-7393-482-4.

No 1294 **Johan Thapper:** Aspects of a Constraint Optimisation Problem, 2010, ISBN 978-91-7393-464-0.

No 1306 **Susanna Nilsson:** Augmentation in the Wild: User Centered Development and Evaluation of Augmented Reality Applications, 2010, ISBN 978-91-7393-416-9.

No 1313 **Christer Thörn:** On the Quality of Feature Models, 2010, ISBN 978-91-7393-394-0.

No 1321 **Zhiyuan He:** Temperature Aware and Defect-Probability Driven Test Scheduling for System-on-Chip, 2010, ISBN 978-91-7393-378-0.

No 1333 **David Broman:** Meta-Languages and Semantics for Equation-Based Modeling and Simulation, 2010, ISBN 978-91-7393-335-3.

No 1337 **Alexander Siemers:** Contributions to Modelling and Visualisation of Multibody Systems Simulations with Detailed Contact Analysis, 2010, ISBN 978-91-7393-317-9.

No 1354 **Mikael Asplund:** Disconnected Discoveries: Availability Studies in Partitioned Networks, 2010, ISBN 978-91-7393-278-3.

No 1359 **Jana Rambusch**: Mind Games Extended: Understanding Gameplay as Situated Activity, 2010, ISBN 978-91-7393-252-3.

No 1373 **Sonia Sangari**: Head Movement Correlates to Focus Assignment in Swedish, 2011, ISBN 978-91-7393-154-0.

No 1374 **Jan-Erik Källhammer**: Using False Alarms when Developing Automotive Active Safety Systems, 2011, ISBN 978-91-7393-153-3.

No 1375 **Mattias Eriksson**: Integrated Code Generation, 2011, ISBN 978-91-7393-147-2.

No 1381 **Ola Leifler**: Affordances and Constraints of Intelligent Decision Support for Military Command and Control – Three Case Studies of Support Systems, 2011, ISBN 978-91-7393-133-5.

No 1386 **Soheil Samii**: Quality-Driven Synthesis and Optimization of Embedded Control Systems, 2011, ISBN 978-91-7393-102-1.

No 1419 **Erik Kuiper**: Geographic Routing in Intermittently-connected Mobile Ad Hoc Networks: Algorithms and Performance Models, 2012, ISBN 978-91-7519-981-8.

No 1451 **Sara Stymne**: Text Harmonization Strategies for Phrase-Based Statistical Machine Translation, 2012, ISBN 978-91-7519-887-3.

No 1455 **Alberto Montebelli**: Modeling the Role of Energy Management in Embodied Cognition, 2012, ISBN 978-91-7519-882-8.

No 1465 **Mohammad Saifullah**: Biologically-Based Interactive Neural Network Models for Visual Attention and Object Recognition, 2012, ISBN 978-91-7519-838-5.

No 1490 **Tomas Bengtsson**: Testing and Logic Optimization Techniques for Systems on Chip, 2012, ISBN 978-91-7519-742-5.

No 1481 **David Byers**: Improving Software Security by Preventing Known Vulnerabilities, 2012, ISBN 978-91-7519-784-5.

No 1496 **Tommy Färnqvist**: Exploiting Structure in CSP-related Problems, 2013, ISBN 978-91-7519-711-1.

No 1503 **John Wilander**: Contributions to Specification, Implementation, and Execution of Secure Software, 2013, ISBN 978-91-7519-681-7.

No 1506 **Magnus Ingmarsson**: Creating and Enabling the Useful Service Discovery Experience, 2013, ISBN 978-91-7519-662-6.

No 1547 **Wladimir Schamai**: Model-Based Verification of Dynamic System Behavior against Requirements: Method, Language, and Tool, 2013, ISBN 978-91-7519-505-6.

No 1551 **Henrik Svensson**: Simulations, 2013, ISBN 978-91-7519-491-2.

No 1559 **Sergiu Rafiliu**: Stability of Adaptive Distributed Real-Time Systems with Dynamic Resource Management, 2013, ISBN 978-91-7519-471-4.

No 1581 **Usman Dastgeer**: Performance-aware Component Composition for GPU-based Systems, 2014, ISBN 978-91-7519-383-0.

No 1602 **Cai Li**: Reinforcement Learning of Locomotion based on Central Pattern Generators, 2014, ISBN 978-91-7519-313-7.

No 1652 **Roland Samlaus**: An Integrated Development Environment with Enhanced Domain-Specific Interactive Model Validation, 2015, ISBN 978-91-7519-090-7.

No 1663 **Hannes Uppman**: On Some Combinatorial Optimization Problems: Algorithms and Complexity, 2015, ISBN 978-91-7519-072-3.

No 1664 **Martin Sjölund**: Tools and Methods for Analysis, Debugging, and Performance Improvement of Equation-Based Models, 2015, ISBN 978-91-7519-071-6.

No 1666 **Kristian Stavåker**: Contributions to Simulation of Modelica Models on Data-Parallel Multi-Core Architectures, 2015, ISBN 978-91-7519-068-6.

No 1680 **Adrian Lifa**: Hardware/Software Codesign of Embedded Systems with Reconfigurable and Heterogeneous Platforms, 2015, ISBN 978-91-7519-040-2.

No 1685 **Bogdan Tanasa**: Timing Analysis of Distributed Embedded Systems with Stochastic Workload and Reliability Constraints, 2015, ISBN 978-91-7519-022-8.

No 1691 **Håkan Warnquist**: Troubleshooting Trucks – Automated Planning and Diagnosis, 2015, ISBN 978-91-7685-993-3.

No 1702 **Nima Aghaee**: Thermal Issues in Testing of Advanced Systems on Chip, 2015, ISBN 978-91-7685-949-0.

No 1715 **Maria Vasilevskaya**: Security in Embedded Systems: A Model-Based Approach with Risk Metrics, 2015, ISBN 978-91-7685-917-9.

No 1729 **Ke Jiang**: Security-Driven Design of Real-Time Embedded System, 2016, ISBN 978-91-7685-884-4.

No 1733 **Victor Lagerkvist**: Strong Partial Clones and the Complexity of Constraint Satisfaction Problems: Limitations and Applications, 2016, ISBN 978-91-7685-856-1.

No 1734 **Chandan Roy**: An Informed System Development Approach to Tropical Cyclone Track and Intensity Forecasting, 2016, ISBN 978-91-7685-854-7.

No 1746 **Amir Aminifar**: Analysis, Design, and Optimization of Embedded Control Systems, 2016, ISBN 978-91-7685-826-4.

No 1747 **Ekhiotz Vergara**: Energy Modelling and Fairness for Efficient Mobile Communication, 2016, ISBN 978-91-7685-822-6.

No 1748 **Dag Sonntag**: Chain Graphs – Interpretations, Expressiveness and Learning Algorithms, 2016, ISBN 978-91-7685-818-9.

No 1768 **Anna Vapen**: Web Authentication using Third-Parties in Untrusted Environments, 2016, ISBN 978-91-7685-753-3.

No 1778 **Magnus Jandinger**: On a Need to Know Basis: A Conceptual and Methodological Framework for Modelling and Analysis of Information Demand in an Enterprise Context, 2016, ISBN 978-91-7685-713-7.

No 1798 **Rahul Hiran**: Collaborative Network Security: Targeting Wide-area Routing and Edge-network Attacks, 2016, ISBN 978-91-7685-662-8.

No 1813 **Nicolas Melot**: Algorithms and Framework for Energy Efficient Parallel Stream Computing on Many-Core Architectures, 2016, ISBN 978-91-7685-623-9.

No 1823 **Amy Rankin**: Making Sense of Adaptations: Resilience in High-Risk Work, 2017, ISBN 978-91-7685-596-6.

No 1831 **Lisa Malmberg**: Building Design Capability in the Public Sector: Expanding the Horizons of Development, 2017, ISBN 978-91-7685-585-0.

No 1851 **Marcus Bendtsen**: Gated Bayesian Networks, 2017, ISBN 978-91-7685-525-6.

No 1852 **Zlatan Dragisic**: Completion of Ontologies and Ontology Networks, 2017, ISBN 978-91-7685-522-5.

No 1854 **Meysam Aghighi**: Computational Complexity of some Optimization Problems in Planning, 2017, ISBN 978-91-7685-519-5.

No 1863 **Simon Ståhlberg**: Methods for Detecting Unsolvable Planning Instances using Variable Projection, 2017, ISBN 978-91-7685-498-3.

No 1879 **Karl Hammar**: Content Ontology Design Patterns: Qualities, Methods, and Tools, 2017, ISBN 978-91-7685-454-9.

No 1887 **Ivan Ukhov**: System-Level Analysis and Design under Uncertainty, 2017, ISBN 978-91-7685-426-6.

No 1891 **Valentina Ivanova**: Fostering User Involvement in Ontology Alignment and Alignment Evaluation, 2017, ISBN 978-91-7685-403-7.

No 1902 **Vengatanathan Krishnamoorthi**: Efficient HTTP-based Adaptive Streaming of Linear and Interactive Videos, 2018, ISBN 978-91-7685-371-9.

No 1903 **Lu Li**: Programming Abstractions and Optimization Techniques for GPU-based Heterogeneous Systems, 2018, ISBN 978-91-7685-370-2.

No 1913 **Jonas Rybing**: Studying Simulations with Distributed Cognition, 2018, ISBN 978-91-7685-348-1.

No 1936 **Leif Jonsson**: Machine Learning-Based Bug Handling in Large-Scale Software Development, 2018, ISBN 978-91-7685-306-1.

No 1964 **Arian Maghazeh**: System-Level Design of GPU-Based Embedded Systems, 2018, ISBN 978-91-7685-175-3.

No 1967 **Mahder Gebremedhin**: Automatic and Explicit Parallelization Approaches for Equation Based Mathematical Modeling and Simulation, 2019, ISBN 978-91-7685-163-0.

No 1984 **Anders Andersson**: Distributed Moving Base Driving Simulators – Technology, Performance, and Requirements, 2019, ISBN 978-91-7685-090-9.

No 1993 **Ulf Kargén**: Scalable Dynamic Analysis of Binary Code, 2019, ISBN 978-91-7685-049-7.

FACULTY OF SCIENCE AND ENGINEERING

Linköping Studies in Science and Technology, Dissertation No. 2218, 2022
Department of Computer and Information Science

Linköping University
SE-581 83 Linköping, Sweden

www.liu.se

LINKÖPING
UNIVERSITY